Anno II - N. 16/17 - 25 dicembre 1987 - Sped. Abb. Post. Gr. III/70 - Distr. MePe

personal computer

N.16/17-L.3500

COMMODORE

PASSEGGIANDO TRA I BANCHI

LE AVVENTURE DI AMIGA

sinclair COMPUTER

SUPER WINDOW

LO SPECTRUM + 3

METTI IL TURBO NEL QL

IL PROBLEMA DELLE PROVE RIPETUTE

SALVATAGGIO SU NASTRO DI FINESTRA VIDEO

UN CONVERTITORE ANALOGICO-DIGITALE

ATARI

IL MENU DELL'ST

STARGLIDER

VIDEO A 160 COLONNE

MSX

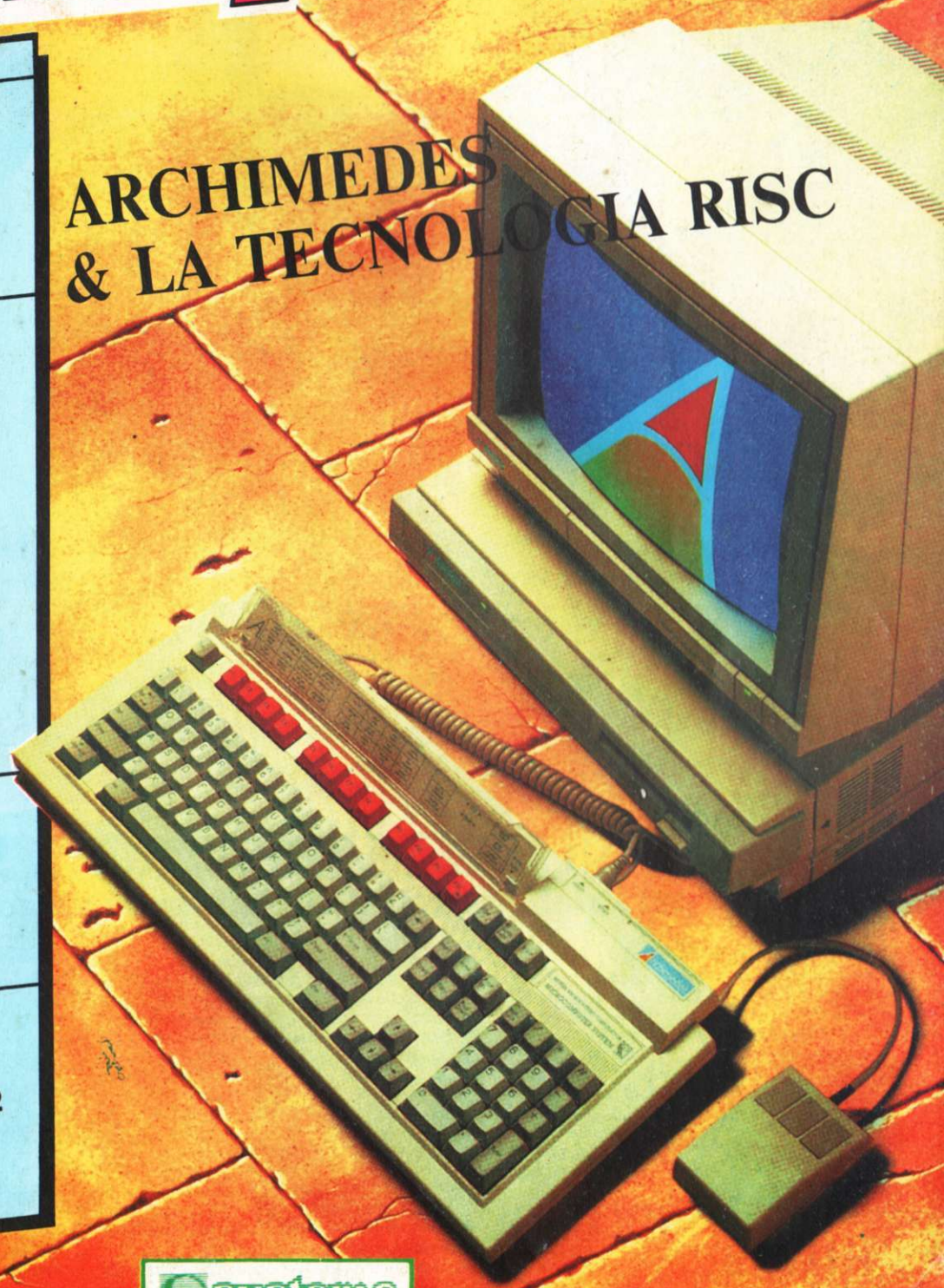
L'ASSEMBLER

64 COLONNE IN SCREEN 2

AUTOLOADER

MUTUI

ARCHIMEDES & LA TECNOLOGIA RISC



Systems

PHILIPS PERSONAL COMPUTER

IL NUOVO VIDEOPROCESSORE
per l'elaborazione delle immagini videoregistrate

NMS

PERSONAL COMPUTER NMS 8280

Linea professionale con tastiera separata dotata di tastierino numerico - 256 KB memoria Ram - consolle incorporante due unità disk da 3,5" da 720 Kb formattati - digitalizzatore - mixer audio/video - eccellente risoluzione grafica (80 colonne - 512x212 punti 256 colori).

LE POSSIBILITÀ CREATIVE

Un programma grafico professionale e un Mouse dati a corredo con il Computer consentono di personalizzare i propri filmati video-registrati attraverso una serie di meravigliosi effetti grafici:

- sovrapposizione tra immagine video e immagine computer;
- 6 effetti wipe di sostituzione di immagine;
- animazione e titolazione;
- digitalizzazione e manipolazione di immagini video;
- stampa e memorizzazione di immagini su disco.

LE APPLICAZIONI PROFESSIONALI

Nonostante le caratteristiche particolarmente orientate alla gestione video l'NMS 8280 resta a tutti gli effetti un personal computer vero e proprio. Il catalogo software è ricco di programmi applicativi professionali. A corredo viene fornito un pack age integrato comprendente video scrittura, gestione archivi, foglio di calcolo elettronico, grafica finanziaria, agenda appuntamenti, pianificatore di risorse, sistema operativo MSX-DOS.

PHILIPS

DIRETTORE RESPONSABILE
Agostina Ronchetti

SEGRETERIA DI REDAZIONE
Maura Ceccaroli

IMPAGINAZIONE & MONTAGGIO
Arturo Ciaglia,
Gabriella Galbusera

REDAZIONE
Fabio Berno, Alessandro de Simone, Valerio Ferri, Luca Lentati, Michele Maggi, Davide Serioli

COLLABORATORI
Damiano Bolla, Luigi Callegari, Sergio Camici, Lorenzo Carrara, Gianluca Carri, Massimo Cellini, Sandro Certi, Andrea Gallo, Monica Fumagalli, Mariangela Guardione, Luca Manfren, Mauro Massetti, Giovanni Mellina, Riccardo Mucciarelli, Marco Mussini, Gianluca Padovani, Antonio Pastorelli, Arturo Pizzuto, Roberto Previtera, Marco Roscio, Franco Toldi, Renzo Zonin, Lorenzo & Roberto Zubani

DIREZIONE, REDAZIONE & AMMINISTRAZIONE
Viale Famagosta, 75
20142 Milano - Tel. 02-8467348

DIFFUSIONE & ABBONAMENTI
Paola Bertolotti

COMPOSIZIONE & FOTOLITO
Systems Editoriale s.r.l.

STAMPA
Stampato da Systems Editoriale/Litografica Srl Busto A.

DISTRIBUZIONE
Messaggerie Periodici S.p.A.
Viale Famagosta, 75 - 20142 Milano
Sped. Abb. post. Gr. III/70

PUBBLICITA':
Milano: Leandro Nencioni (direttore vendite), Guido Agosti, Giorgio Ruffoni, Claudio Tidone. - V.le Famagosta, 75 - 20142 Milano - Tel. 02-8467348
Emilia Romagna: Spazio E. P.za Roosevelt, 4 - 40123 Bologna Tel. 051-236979
Lazio & Campania: Spazio Nuovo Via P. Foscari, 70 - 00139 Roma - Tel. 06-8109679
SEGRETERIA: Paola Bertolotti

SYSTEMS EDITORIALE srl
Reg. Nazion. Stampa n. 01500 vol. 15 fg. 793
Autorizz. Trib. di Milano n. 68 dell'11-02-1984

TARIFFE & ABBONAMENTI
Una copia L. 3.500 (arretrati il doppio). Abbonamenti: vedere in ultima pagina. Gli abbonamenti e le richieste di arretrati vanno indirizzati a:
PERSONAL COMPUTER, V.le Famagosta, 75 - 20142 Milano, sotto forma di assegno bancario non trasferibile o di versamento su c/c postale n. 37952207. Per i cambi di indirizzo, specificare chiaramente sia il nuovo che il vecchio recapito e allegare L. 500.

Personal computer

sommario

rubriche attualità

04 *Archimedes & la tecnologia Risc*

COMMODORE

06 *Passeggiando tra i banchi*

12 *Le avventure di Amiga*

sinclair COMPUTER

18 *Super window*

20 *Gare con l'arco*

21 *Mailbox*

23 *Lo Spectrum +3*

25 *Metti il turbo al tuo QL*

27 *Othello*

27 *Delete files*

27 *Calcolo balistico*

28 *Il problema delle prove ripetute*

77 *QL bullettin board*

80 *Salvataggio su nastro di finestra video*

81 *Hackers & Crackers*

83 *Un convertitore analogico digitale per lo Spectrum*

ATARI

88 *Il menu dell'ST*

89 *Starglider*

91 *Zetagraf*

92 *Video a 160 colonne*

MSX

93 *L'assembler*

97 *Zig Zag*

98 *64 colonne in screen 2*

99 *Autoloader*

100 *Mutui*

100 *Quiz*

101 *Usa & abusa*

102 *Mailbox*

listati

I/29-XLVIII/76

ARCHIMEDES & LA TECNOLOG

All' "Acorn user exhibition" a Londra è stato presentato Archimedes, il nuovo computer della Acorn (gruppo Olivetti), ditta inglese di Cambridge famosa in tutto il mondo per i suoi home computers in uso nelle scuole. Destinato a concorrere con i vari PC, Archimedes è un 32-bit a tecnologia RISC (Reduced Instraction Set Computer) capace di ben 4 mips (Million Instructions Per Second) e circa 10 volte più veloce di un IBM PC, di un Apple Macintosh o di un Atari ST (nel volantino pubblicitario sono inclusi i risultati di vari test eseguiti da un'illustre rivista inglese del settore). E' fornito di 4096 colori diversi, dei quali 256 possono essere presenti sullo schermo allo stesso istante; ha un suono stereo digitale a otto voci, da mezzo fino a quattro megabytes di RAM, 512K di ROM e usa uno o due floppy discs da 3.5", capaci di memorizzare un megabyte. E' disponibile anche un hard disc da 40 megabytes.

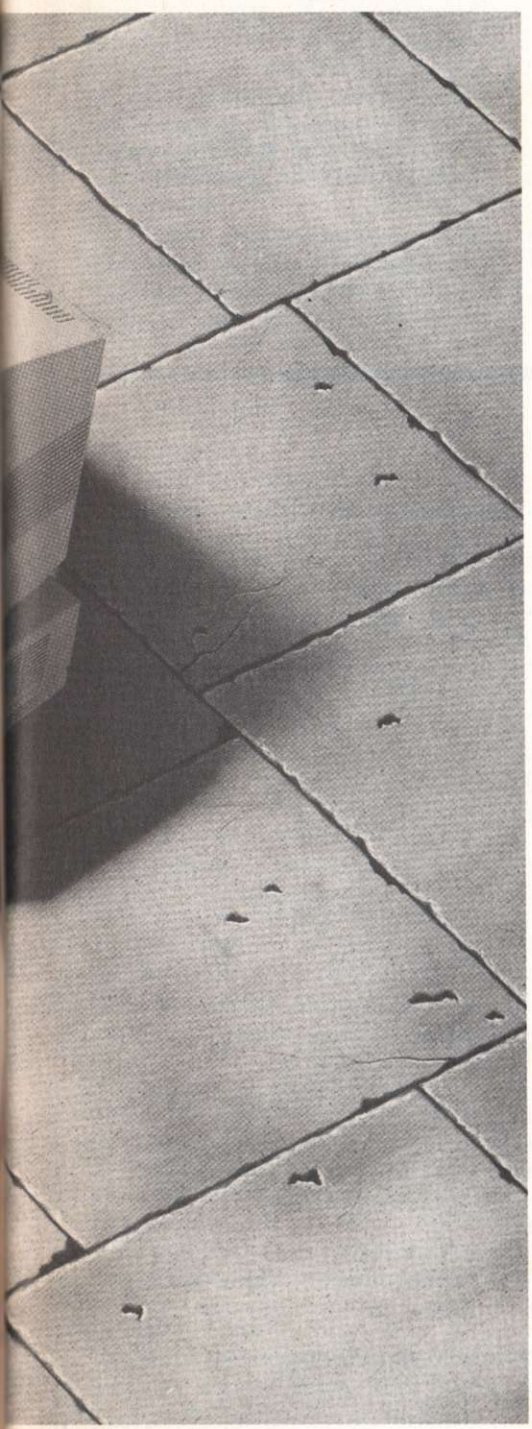
Il seguente programma viene completato in meno di 3 secondi mentre ad uno Spectrum occorre circa un minuto (non si dovrebbero proprio paragonare due macchine di livelli tanto diversi):

```
10 FOR A=1 TO 1000
20 LET X=SIN (A)
30 NEXT A
```

In Italia verrà probabilmente importato dalla Ricordi.

A RISC

a cura di Andrea Gallo



Il termine RISC è l'abbreviazione di Reduced Instruction Set Computer ed indica un particolare tipo di computers, destinati nei prossimi anni a superare in prestazioni e minor costo di produzione gli attuali personal computers.

Vediamo innanzitutto quali sono le differenze tra questi due generi. Sebbene entrambi mirino a eseguire i programmi più complessi nel minor tempo possibile, essi hanno una struttura interna differente, ispirata dalle possibilità tecnologiche ed economiche mutate nel corso degli anni.

Fino a poco tempo fa, infatti, i chips di memoria vuoti costavano molto per cui i computers che usiamo ancora oggi sono stati costruiti in modo tale da avere il numero maggiore di istruzioni possibile per scrivere programmi complessi in modo più breve. Per esempio sullo Z80 abbiamo istruzioni quali LDIR, CPIR, INIR, ecc., che muovono larghi blocchi di dati occupando solo 2 bytes (circa 12 se consideriamo le istruzioni per indirizzare i registri e quindi preparare lo spostamento) invece dei 40 o 50 necessari a scrivere la routine per compiere tale compito istruzione per istruzione.

Al giorno d'oggi, dato che i prezzi dei chips vuoti sono notevolmente calati mentre è costosa la costruzione di complessi processori, utilizzando la tecnologia RISC si tende a progettare computers con il minor numero di istruzioni possibile anche a costo di scrivere programmi maggiori, non essendo più un problema la memoria vuota. Rifacendoci all'esempio precedente, è meglio avere un microprocessore sprovvisto dell'istruzione LDIR, quindi più semplice e veloce, e scrivere la routine utilizzando 30 o 40 bytes in più.

La prima ditta a progettare tale tipo di microprocessori fu la IBM che nel 1975 iniziò a disegnare il Project 801, lavoro reso noto solo nel 1982.

Nel 1981 i ricercatori dell'Università della California realizzarono il RISC1, introducendo anche il termine di tecnologia RISC. Il loro computer, sebbene ancora ai primi passi, era più veloce di un microprocessore Motorola M68000 (simile a quello usato dal QL).

Nel 1981 anche la Acorn si interessò al progetto e successivamente presentò la Acorn Cambridge Workstation.

Nel Settembre 1983 i progetti del RISC1 vennero resi noti e la Acorn cominciò a disegnare il progetto ARM (Acorn Risc Machine) da cui è nato Archimedes.

Il microprocessore ARM è dotato di una grande versatiilità: ogni istruzione può avere associata una condizione; infatti dei 32 bits di ogni byte 4 sono utilizzati per definire la condizione in cui eseguire l'istruzione e i rimanenti bits definiscono il codice operativo dell'istruzione, i registri interessati e valori numerici o indirizzi di memoria. E' interessante notare che tutte le istruzioni occupano un byte e che la memoria viene indirizzata solo direttamente ovvero non si hanno istruzioni simili alla CP (HL) sullo Z80. Inoltre i vari "jump" e "branch" sono ad indirizzi relativi, avendo un "range" di + o - 64K, il che rende un programma in l/m completamente rilocabile.

Ecco un esempio in ARM assembler con i rispettivi Z80 e BASIC:

ARM	Z80	BASIC
CMP Rn, #p	LD A,(Rn)	IF Rn=p OR Rm=q THEN GOTO Label
CMPNE Rm, #q	CP p	...
BEQ Label	JR Z,Label	
...	LD A,(Rm)	
	CP q	
	JR Z,Label	
	...	

Una delle cose più interessanti è che il microprocessore ARM lavora ad una velocità tale per cui mentre esegue un'istruzione, interpreta la seguente e allo steso tempo cerca la successiva, il tutto in una sola istruzione di clock, il quale lavora a 4 o 8 Mhz.

PASSEGGIANDO TRA I BANCHI

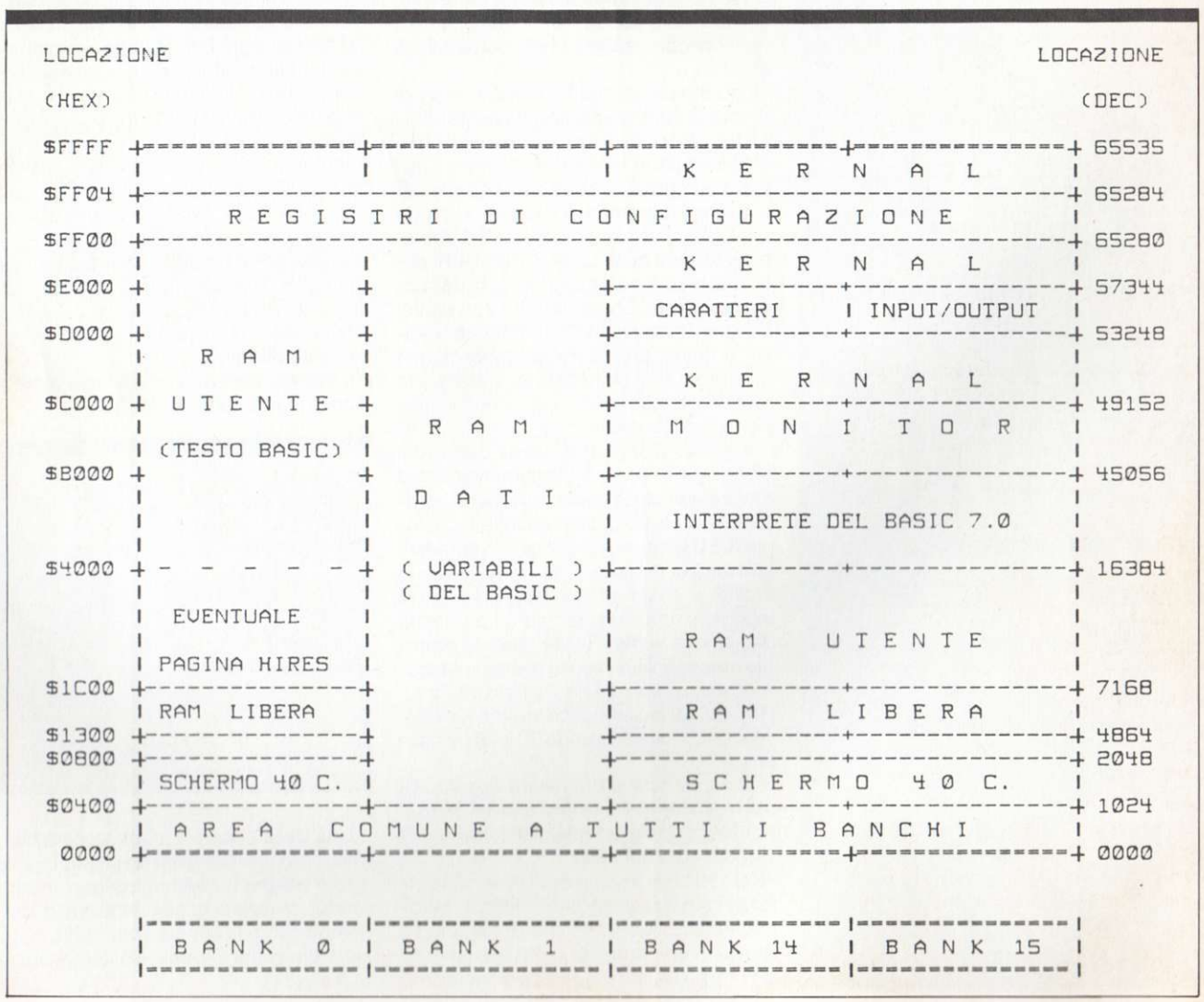
di Domenico Pavone

Come sfruttare in pratica le diverse configurazioni di memoria del C/128

Per chi possiede un Commodore 128, uno dei momenti più gratificanti è certamente quello dell'accensione della macchina, quando le prime righe dello schermo ci informano di avere a disposizione una quantità di memoria libera ben al di là dei 64 K (65536 byte), estremo limite fisico dei microprocessori a 8 bit con indirizzamento a 16 bit.

Perchè limite fisico? Bè, è (quasi) semplice. Indirizzamento a 16 bit significa che è possibile localizzare una certa cella di memoria tramite una coppia di byte (2x8 = 16 bit), per cui il numero massimo di locazioni gestibile da questo tipo di processori è di 2 elevato alla sedicesima potenza, cioè 65536. Se volete approfondire l'argomento, procuratevi il numero 8 del lontano 1984 di questa rivista, oppure spulciate l'inserto pubblicato sul n.45.

Il microprocessore adottato dal C/128 è il microcircuito 8502, niente di più che una evoluzione dell'unità che fa "muovere" il buon vecchio



C/64. Quindi proprio un 8 bit con tanto di indirizzamento a 16 bit e con lo stesso limite di gestione di 65536 byte, ma inserito in una struttura tale che gli consenta di accedere, in rapidissima successione, a più di una unità di memoria da 64 K.

Queste unità sono definite banchi, e la struttura, che ne consente la gestione, è la Memory Management Unit, abbreviata MMU. Se avete dato un'occhiata al manuale del computer, saprete già che i banchi disponibili sono 16 e che l'interprete Basic 7.0 possiede l'istruzione BANK che permette di accedere liberamente ad uno di essi; che cosa farne, però, sembra essere top secret.

Ciò che ci proponiamo è proprio di utilizzare, in pratica, le possibilità offerte da una tale organizzazione interna del computer, ma prima è necessario approfondire la conoscenza su banchi e MMU.

I banchi

Anzitutto occorre precisare (come si deduce dal manuale) che dei 16 banchi ben 6 sono da escludere (i n. 2, 3, 6, 7, 10 e 11) in quanto finalizzati ad eventuali estensioni di memoria. Dei rimanenti, quelli che più comunemente vengono sfruttati nella programmazione, e che servono per capire come "destreggiarsi", sono quattro (n.0, 1, 14, 15), e li potete vedere schematizzati in figura 1, rappresentati in senso verticale. Ogni banco ha una sua precisa conformazione, con i 64 K suddivisi in varie aree con varie funzioni. Di ognuna di queste aree sono indicate le locazioni di inizio e di fine, a sinistra in esadecimale e a destra in decimale.

E' bene ricordare che il C/128 è fornito di un ottimo Monitor nonchè dell'istruzione DEC, per cui vi invitiamo a farne uso se non gradite il sistema esadecimale, al quale, d'ora in poi, faremo riferimento; per quanto riguarda le istruzioni del Monitor, oltre il manuale potete anche consultare il n.34 di CCC a pag.76.

Una prima differenza fondamentale è che esistono due tipi di Ram (ovviamente solo dal punto di vista logico), definite nel manuale come Ram 0 e Ram 1, che in figura, per maggiore chiarezza, sono indicate come Ram Utente e Ram Dati. Se si considera la programmazione in Basic, nella prima può essere memorizzato il testo vero e proprio del programma, mentre nella seconda vengono immagazzinate tutte le variabili. Sempre facendo riferimento alla figura 1, potete notare come nei Bank 0, 14 e 15 sia presente solo Ram 0 (cioè utente), mentre nel Bank 1 sia presente solo Ram 1. E, come si può dedurre dal preponderante spazio riservato alla Ram, il Bank 0 viene utilizzato dal sistema per contenere il testo dei programmi Basic, il Bank 1 le sue variabili.

L'area definita come Ram Libera è sempre Ram 0, ma posta al di sotto dell'inizio del testo Basic, quindi esente da interferenze e ideale per allocarvi subroutine in linguaggio macchina (l.m.).

Importante da considerare è che se digitate un programma, questo verrà inserito in Bank 0, ma sembra "presente" anche in tutti gli altri banchi che contengono Ram 0 nelle stesse locazioni in cui è memorizzato il programma originale.

Per essere più chiari, se in Bank 0 c'è un programma Basic oppure l.m. che occupa le locazioni da $1C01 (inizio del Basic) a $1FFF, questo lo ritroverete nelle stesse locazioni di Bank 14 e di Bank 15 destinate a Ram Utente (e viceversa). Se, invece, lo stesso programma si trova allocato a partire da $4000 (inizio del Basic se si imposta la pagina grafica) sempre di Bank 0, ovviamente in Bank 14 e 15 non potrà esserci, in quanto le stesse locazioni sono occupate dall'interprete Basic.

Le Rom necessarie al sistema per operare risiedono nei banchi 14 e 15; l'unica differenza tra i due è rilevabile nell'area da $D000 a $E000, dove si trova il generatore di caratteri nel banco 14 e le funzioni di input/output (I/O) nel banco 15. Per default il computer si trova in quest'ultima configurazione.

Prima di passare agli "smanettamenti", è indispensabile notare come le locazioni numerate da 0 a $0400 siano definite come "area comune", il che significa che vengono condivise da tutte le possibili configurazioni, indipendentemente anche dal tipo di Ram; vedremo più avanti l'importanza di questa area. Un'altra piccola area comune a tutti i banchi è quella da $FF00 a $FF04, che contiene la copia di 5 registri della MMU. Già, perchè la MMU cui si è accennato in precedenza altro non è che un gruppo di registri (12 per l'esattezza) locati da $D500 a $D50B, che consentono di manipolare le configurazioni di memoria in modo da adattarle alle più svariate esigenze.

Alcuni di questi registri, e più precisamente i primi 5, servono per cambiare banco. Le locazioni di cui stiamo parlando si trovano in Bank 15, ma per consentirne l'accesso da qualunque configurazione, ne esiste una copia, appunto, nell'area comune da $FF00 in poi.

Abbiamo quindi due modi per modificare la conformazione di memoria in cui si vuole operare: uno, da Basic, tramite il comando Bank; l'altro, più versatile, inserendo certi valori nella locazione $FF00. Ma attenzione: se il numero che si pone dopo Bank (in Basic) indica direttamente il banco cui si vuole accedere, diverso sarà il valore da inserire nel registro di configurazione FF00. Il perchè lo diremo in seguito; per adesso limitatevi a digitare il breve listato 1.

Tanto per entrare nella giusta ottica da seguire quando si realizza un programma, ricordiamo che, all'accensione del computer, ci si trova in Bank 15, per cui la SYS di riga 40 manderà in esecuzione una routine del Kernal (controllate in figura 1). La routine si limita a fornire il valore da inserire in $FF00 a seconda del banco che si vuole impostare; per attivarla è necessario inserire nel registro X il numero del banco (variabile X nel programma), mentre il valore desiderato lo si potrà leggere nell'accumulatore, il cui contenuto è conservato nella locazione 6 di pagina zero. Impartendo RUN al programma, otterrete una tabella di valori (in decimale ed esadecimale) per i sedici banchi: a voi il compito di passarli eventualmente su stampante o copiarli su carta per averli più facilmente sott'occhio.

Ma vediamo di concretizzare quanto sinora detto. Anzitutto occorre precisare che se si programma in Basic "stretto", cioè senza neanche una poke o una peek (figuriamoci poi una Sys!), si può anche ignorare l'istruzione Bank, in quanto è il sistema operativo che si occupa di organizzare testo, variabili, pagina grafica nei rispettivi banchi senza la necessità di interventi esterni. Ma se le vostre esigenze sono più sofisticate, proseguite la lettura enunciando una regola fondamentale

nella programmazione del C/128:

Se si compie una qualsiasi operazione riferita a una certa locazione di memoria (lettura, scrittura o esecuzione di routine in l.m.), occorre sempre tenere bene a mente in quale conformazione di banco si stia operando.

Questo vale anche se ci si trova in ambiente Basic. Per averne un'idea pratica, digitate e salvate su il programma 2, che visualizzerà il contenuto del registro di configurazione FF00 dopo il Run e dopo ogni cambiamento di banco. Dando un'occhiata alla tabella che vi ha fornito il primo programma, saprete in che banco ci si trova dopo lo start del programma.

Se avete effettuato il salvataggio su nastro o disco, ora resettate il computer e caricate nuovamente il programma per essere sicuri di trovarci nelle condizioni di default del sistema. Dopo il Run noterete come il contenuto del registro sia 0, cioè ci si trova in banco 15, e questo anche se sta "girando" un programma Basic. Il fatto che il programma sia depositato in Bank 0, infatti, non vuol dire che ci si trovi in quella configurazione, ma solo che il sistema operativo preleverà da lì le istruzioni del testo, le manderà in esecuzione, per poi tornare alla configurazione selezionata, in questo caso Bank 15.

Premendo un tasto, ogni volta si provocherà un cambiamento di banco e quindi una variazione del contenuto di $FF00. Per avere una conferma di quanto detto, provate ad arrestare il programma premendo il tasto STOP mentre ci si trova in un banco diverso dal 15. Annotate il contenuto di $FF00 e date nuovamente il Run. Il valore che apparirà sarà lo stesso che avete annotato, il che, in altre parole, significa che il sistema è rimasto nella stessa configurazione dell'ultimo comando Bank, e qui rimarrà fino a nuovo ordine.

Per non creare equivoci, ricordiamo che il "trovarsi" in un certo banco non significa che è operativo solo quello, ma che il sistema vi ritorna dopo avere usato le sue routine di bank 15, dopo aver prelevato il testo Basic da bank 0, ecc.

Se tutto quanto vi risulta chiaro, a questo punto siamo pronti per vedere come sfruttare la memoria disponibile che, rinunciando ad operare solo in Basic, diventa veramente immensa. Per questo linguaggio, infatti, sono disponibili i banchi 0 e 1, ma, come abbiamo detto, solo il banco 0 è riservato al testo (non è che sia poco!), mentre se si ci si muove in l.m, anche solo a livello di subroutine, praticamente si può utilizzare a proprio piacimento tutta la memoria sia di banco 0 che di banco 1; rispettando, ovviamente, le regole imposte dalla macchina, sulle quali ora ci soffermeremo.

Negli esempi che seguiranno, "giocheremo" con la scritta che appare quando si accende il computer, i cui caratteri sono immagazzinati nell'area riservata all'interprete del Basic dalla locazione $41BF alla locazione $4250; di quali banchi? Se non lo sapete, vi consiglio di rileggere tutto daccapo.

Tanto per dare una prima occhiata, spegnete e riaccendete il computer, quindi digitate il listato 3, salvatelo e lanciate il programma. L'ultima parte della linea 30 serve solo a eliminare dei caratteri che "sporcano" lo schermo. Fin qui niente di nuovo, in quanto il programma opera mentre il sistema si trova in bank 15 e quindi non esiste alcun problema nel leggere le locazioni della Rom interprete.

Proviamo ora a movimentare un po' le cose, facendo riferimento al listato 4 (da digitare e salvare). Qui anzitutto si leggono i caratteri della scritta e li si copiano in Bank 1 a partire dalla locazione $2000; poi, per essere stampati, vengono prelevati da quest'ultimo e non più dalla Rom di banco 15. Come vedete, da Basic non è poi molto difficile operare in lettura/scrittura tra un banco e l'altro, in questo esempio l'unico punto cui prestare attenzione è in quale banco ci si trova prima del Print di linea 60, e questo vi viene mostrato tramite il solito valore di $FF00 prima dello STOP di riga 40. Dopo l'interruzione potete subito dare il classico Cont, ma vi consiglierei prima di verificare quanto è avvenuto entrando in Monitor e digitando M 12000. Se tutto è in ordine, dovreste trovare la nostra scritta. A riprova di quanto sperimentato, dopo che il programma ha concluso, provate con RUN 40; la scritta è ancora presente nelle locazioni di Bank 1, ma appariranno solo caratteri fortuiti, in quanto verranno letti dalla Ram di banco 15.

Per la cronaca, l'operazione di trasferimento di memoria da un banco all'altro è possibile effettuarla direttamente tramite il Monitor; saprete già che la cifra prima del numero di locazione rappresenta proprio il banco, per cui un comando...

Monitor
***T** F41BF F4250 12000*

...eseguirà lo stesso lavoro che svolgono le righe da 10 a 30 dell'ultimo listato.

Una parentesi prima di proseguire: nell'esempio appena proposto, come talvolta in quelli successivi, si fa un uso piuttosto disinvolto delle locazioni di Bank 0 e Bank 1 per rendere il più chiaro possibile l'argomento che stiamo trattando. Se però volete cimentarvi in operazioni simili, non bisogna dimenticare di "proteggere" le zone che si vogliono adoperare dall'influenza del Basic modificando il valore di alcuni suoi puntatori. Eccovene il dettaglio, in notazione esadecimale:

$2D - $2E: Inizio del testo del programma.
$1210-$1211: Fine del testo Basic.
$2F - $30: Inizio area delle variabili.
$31 - $32: Fine variabili e inizio vettori e matrici.
$33 - $34: Fine vettori e matrici.
$35 - $36: Inizio area stringhe.
$39 - $3A: Fine area stringhe.

Sul come operare in merito non è il caso di soffermarsi in quanto l'argomento meriterebbe un trattato, e comunque è stato spesso affrontato sulla rivista.

In linguaggio macchina

Ma veniamo adesso alla parte più delicata della nostra escursione tra i banchi del C/128, quella che riguarda l'uso di routine in linguaggio macchina. Cercheremo di essere il più chiari possibile, ma è ovvio come, da questo punto in poi, si pretenda una sia pur minima conoscenza dell'Assemby. Come applicazione pratica continueremo ad usare il logo di start del Basic 7.0 (la solita scritta, per intenderci) limitandoci alla prima riga, ma per evidenziare le varie possibilità di interscambio tra i banchi faremo sì che questa appaia in permanenza sullo schermo tramite una manipolazione dell'interrupt.

Come forse già saprete (vedi anche

CCC n.13 ed inserto del n. 46), questa tecnica consiste nel modificare i valori di un vettore del sistema in maniera da "puntare" ad una nostra routine, la quale verrà eseguita ogni sessantesimo di secondo, cioè ogni volta che il computer eseguirà una interruzione per svolgere una serie di operazioni, tra cui la scansione della tastiera. Il vettore in questione si trova nelle locazioni $314 e $315 (le stesse del C/64 e addirittura del Vic-20), e normalmente punta a $FA65 del Kernal. Per ottenere il nostro scopo è necessario inserire, in tali locazioni, (nel formato basso/alto) l'indirizzo di partenza della nostra routine, la quale dovrà concludersi con un salto a $FA65.

Torniamo ora alla tastiera. Digitate e salvate il programma del listato 5, il quale si limita ad allocare e mandare in esecuzione la routine che potete vedere commentata nel disassemblato 1. Questa è allocata a partire da $1300, nella Ram di tipo 0 non raggiungibile dal testo Basic, e potremmo definirla la versione "di base", in quanto, per svolgere il suo compito, non necessita di alcun mutamento di banco. Se date il Run al programma, in alto sullo schermo appariranno i caratteri della prima linea del logo, e vi rimarranno anche se se premete i tasti Shift+Clr. Tutto anche troppo semplice, vero? Bene, vediamo di complicare un po' le cose.

Supponiamo che, per non meglio precisate esigenze, dobbiamo allocare altrove la routine, per esempio da $C000 del banco 0. I problemi che si pongono sono i seguenti:

- *L'indirizzo del vettore di interrupt deve puntare ad una locazione di Bank 0.*
- *La routine vera e propria, pur operando da bank 0, deve prelevare i caratteri del logo dal banco 15.*
- *Ultimate le operazioni, occorre saltare a $FA65, ma di banco 15.*

Dopo le precedenti "dissertazioni" sul registro di configurazione FF00 le cose non sembrerebbero poi così difficili, ma interviene un'altra regola di primaria importanza: non è possibile, all'interno di una routine l.m, cambiare banco alterando il valore di $FF00, se nel banco di arrivo non è presente la stessa routine.

Se, quindi, ci limitassimo ad inserire nel nostro programma qualche istruzione tipo...

Routine di sistema per interscambio tra i banchi

INDFET

Preleva un byte da un banco qualunque e lo deposita nell'accumulatore. Per utilizzarla occore effettuare un salto (JSR) a $F7D0 dopo avere settato i seguenti parametri:

- in due locazioni di pagina zero si inserisce l'indirizzo da cui prelevare il byte nel formato basso/alto
- in accumulatore il numero della prima delle due locazioni di pagina zero (se l'indirizzo lo avete messo in $FA e $FB, allora: LDA #$FA)
- nel registro X il numero di banco (quello reale: per banco 1 = LDX #$01)
- nel registro Y un eventuale indice da sommare all'indirizzo di "prelevamento"

Per usare questa routine è necessario portarsi in banco 15 per saltare al Kernal, ma esiste anche un altro modo di utilizzarla:

Si preparano le due celle di pagina zero come sopra, si inserisce il numero della prima in $02AA (non più in accumulatore), in X va inserito l'indice del banco da cui leggere (non il numero di banco, ma il valore che serve per $FF00!), Y svolge la stessa funzione. Il salto di ingresso è a $02A2, e lo potrete effettuare da qualsiasi banco.

INDSTA

E' l'opposto di INDFET: inserisce il contenuto dell'accumulatore in una locazione di qualsiasi banco. L'ingresso è a $F7DA e i parametri da settare sono:

- in A il byte da trasferire
- in due celle di pagina zero l'indirizzo in cui si desidera "pokare" il valore; il numero della prima di queste va messo in $02B9
- in X il numero (reale) di banco
- in Y il solito offset

Anche per questa routine si può "aggirare" l'obbligo di chiamarla da banco 15 semplicemente inserendo in X l'indice del banco invece del suo numero "vero" e saltando a $02AF; gli altri parametri vanno preparati allo stesso modo.

INDCMP

Opera un confronto tra l'accumulatore e il contenuto di una locazione di qualsiasi banco. Ingresso in $F7E3.

- in A ovviamente va il valore da confrontare
- in X il numero di banco (reale)
- in $02C8 il numero della prima delle due locazioni di pagina 0 con l'indirizzo desiderato
- in Y l'offset

Anche qui si può accedere alla routine da un indirizzo più "comodo", che è $02BE; in questo caso, come al solito, in X andrà non il numero reale del banco ma il valore da inserire in $FF00 per commutarlo, il resto non cambia. L'esito del raffronto sarà ovviamente leggibile dal registro di stato, o se si preferisce dalla locazione 5 in cui esso è copiato.

JMPFAR e JSRFAR

Saltano ad una locazione di qualsiasi banco. La prima è ovviamente senza ritorno (JMP) mentre la seconda è la classica chiamata a una subroutine (JSR).

L'indirizzo di accesso al JMPFAR è $02E3, quello a JSRFAR $02CD. Entrambe devono obbligatoriamente essere chiamate da banco 15, o non funzionano. I parametri da settare sono gli stessi, e vanno inseriti nelle locazioni di memoria da 2 a 8 nel seguente ordine:

- numero reale di banco (2)
- byte alto dell'indirizzo (3)
- byte basso dello stesso (4)
- registro di stato (5)
- registri A (6), X (7), Y (8)

I parametri di ritorno di JSRFSR (ovviamente JMPFAR non può averne) si avranno nelle locazioni da 5 a 9 che conterranno, sempre nell'ordine, lo status register, i registri A, X ed Y, e infine lo stack pointer.

LDA #$00
STA $FF00

...per portarci in banco 15, l'unico risultato sarebbe un blocco del sistema, in quanto ovviamente non può mai esserci una nostra routine in Rom. Ma il modo di aggirare il problema c'è, anzi ce ne sono due.

Anzitutto ci viene in soccorso una eccezione alla regola appena enunciata, e questa è rappresentata dall'area comune (locazioni da 0 a $400). In essa si può liberamente operare un salto di banco senza alcuna conseguenza. Per trovare spazi utilizzabili in questo Kbyte, dovrete affidarvi a una buona mappa di memoria, come per esempio quella pubblicata dalla rivista cugina "Personal computer" (n.7, ott. '86).

Un altro grosso aiuto il sistema operativo lo fornisce con alcune sue routine che consentono tutte le operazioni di interscambio tra i banchi, e che vedete descritte nel riquadro a parte.

Vediamo allora come può essere riorganizzata la nostra routine, facendo riferimento al disassemblato 2, che potete copiare direttamente da Monitor o usando il caricatore Basic di listato 6.

Come potete notare dalle istruzioni da 0C000 in poi, nel vettore di interrupt viene inserito l'indirizzo $028D. Questo si trova nell'area comune a tutti i banchi, e, più precisamente, in un buffer del Monitor e del Basic che occupa le locazioni da $200 a $2A1. Qui si specifica il banco 0 (tenete presente i valori della tabella che abbiamo ottenuta in precedenza) e si salta alla sezione principale del programma, presente da $C00D.

Per leggere le locazioni di bank 15 si sfrutta la routine INDFET (vedi riquadro) usando l'ingresso in $2A2; per il salto a $FA65 si usa ancora il "passaggio" in area comune per il cambio di banco ($295).

Non ci resta che affrontare un'ultima possibilità, quella dell'utilizzo della Ram DATI (Ram 1). Le metodiche adottate fin qui sono più che sufficienti per muoversi correttamente anche al suo interno, ma non è superfluo un ulteriore ripasso con la proposta di un nuovo problema.

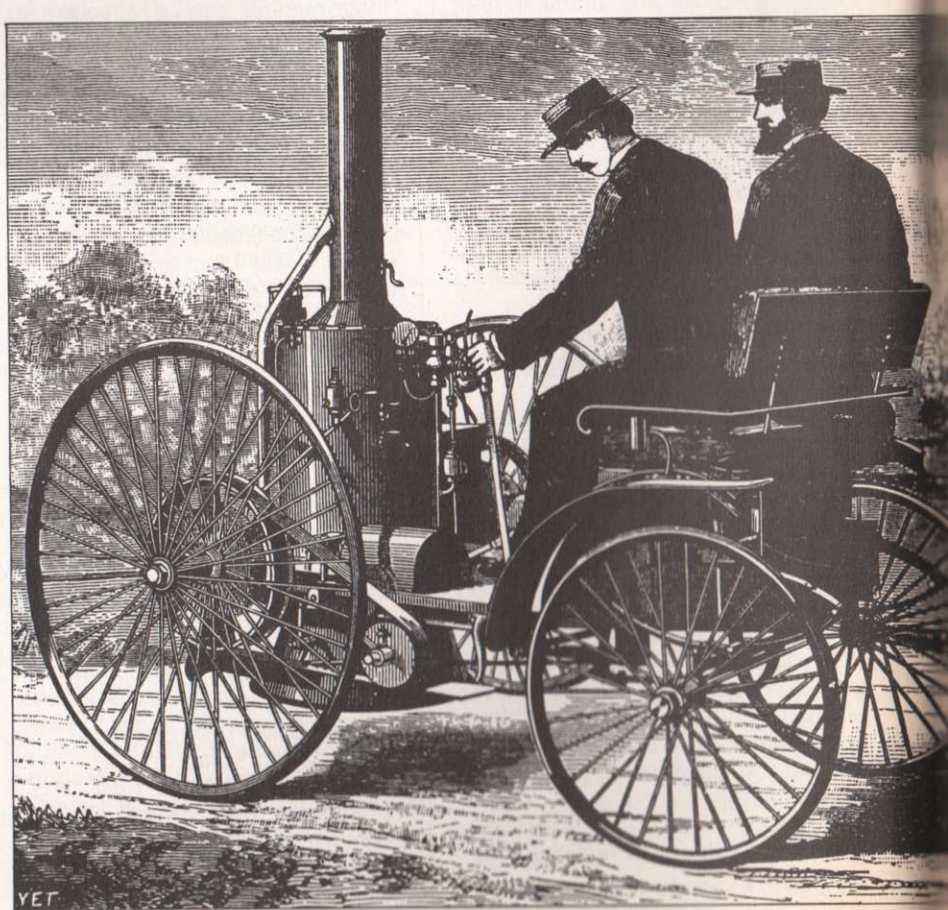
Sempre considerando la stessa routine, supponiamo stavolta di avere in Bank 0 la parte che si occupa di modificare l'interrupt, e in Bank 1 il resto (vedi disassemblato 3 e, per gli "aficionados" del Basic, il relativo caricatore di listato 7.

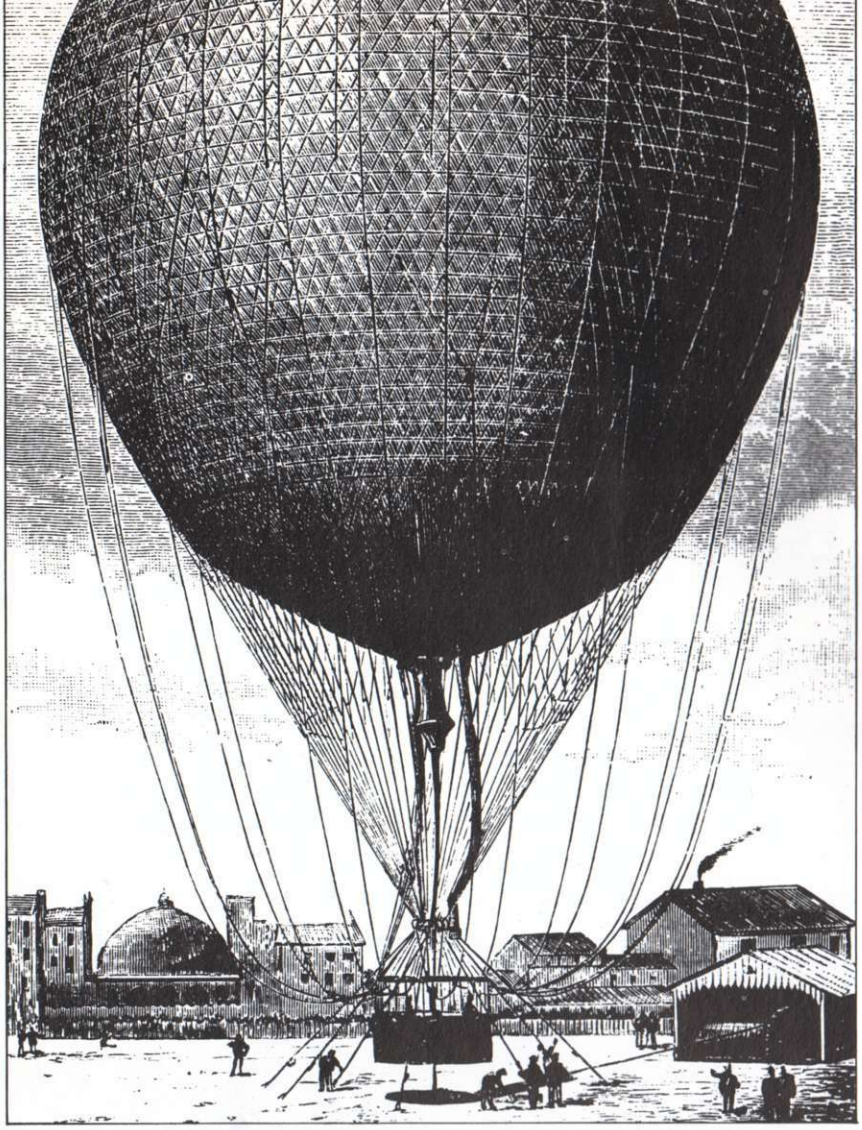
A parte il diverso valore da assegnare a $FF00 per il cambio di banco (istruzione in 0028D), il modo di procedere non cambia, con un'unica eccezione: nel banco 1 non è disponibile l'area dello schermo, per cui, una volta ricevuto in accumulatore un carattere tramite INDFET, sarà necessario consegnarlo alla routine INDSTA per inviarlo in un banco (15 nell'esempio) ove sia presente lo schermo.

Se vi è tutto chiaro, a questo punto dovreste essere in grado di affrontare qualunque problema si ponga nel passare da un banco all'altro della memoria del C/128. Tenete però presente che, nonostante la velocità del linguaggio macchina, lo "switch" comporta pur sempre un certo rallentamento. Per cui spostare grosse aree di memoria (come un'intera schermata hi-res) da un banco all'altro... ahimè, farà tornare alla mente le fosche considerazioni espresse nel già citato inserto del numero 45.

Ma per riscattare, almeno in parte, l'onore del nostro amato C/128, eccovi un'ultima tornata di informazioni che consentono di limitare all'indispensabile i ritardi dovuti a continui salti da un banco all'altro.

Abbiamo visto in precedenza come, per impostare una delle 16 configurazioni possibili di memoria, sia sufficiente inserire nel registro FF00 un certo valore, ricavabile sfruttando la routine del Kernal posta da $F7EC (GETCFG). Ebbene, i sedici valori con i quali abbiamo realizzato la tabella non sono gli unici leciti! E' possibile cioè inserirne altri, a nostra scelta, per determinare una conformazione di memoria "personalizzata"; se, per esempio, vogliamo usare un programma l.m. piuttosto lungo (o comunque allocato in maniera tale da richiedere l'area fino a $C000) che debba effettuare frequenti salti al Kernal, possiamo creare un banco che disponga di Ram fino a $C000 e del Kernal da $C000 in su (fate sempre riferimento alla figura 1). Per ottenere quanto detto, il valore di FF00 dovrà essere conformato considerando la sua struttura binaria, in quanto ogni bit che lo com-





pone modifica una certa area di memoria in accordo con le seguenti modalità:

BIT 0
Controlla l'area da $D000 a $DFFF. Se è posto a zero, questa conterrà l'I/O. Se è posto a 1, vi si troverà il generatore di caratteri (o Ram), a seconda della conformazione dei bit 4 e 5.

BIT 1
Area da $4000 a $8000. Posto a zero, insedierà la Rom bassa del Basic, che, per chiarezza, in figura 1 è illustrata come unica, mentre, in effetti, è divisa in LOW e HI Rom. Settato a 1, la stessa area conterrà solo Ram.

BIT 2 e 3
Locazioni da $8000 a $C000. Se entrambi sono azzerati, saranno presenti la Rom "alta" del Basic più il Monitor. La configurazione binaria 11 selezionerà invece Ram.

BIT 4 e 5
Presiedono alla zona da $C000 a $FFFF. Se il contenuto di entrambi è zero, sarà allocato il Kernal più il generatore di caratteri o l'I/O, a seconda del contenuto del bit 0. Se invece i bit sono tutti e due settati (11), indicheranno solo Ram.

BIT 6 e 7
Selezionano il tipo di Ram. 00 farà sì che tutta la Ram presente sia di tipo 0, mentre con 01 si selezionerà la Ram di tipo 1.

La configurazione cui abbiamo accennato (poche righe fa) come esempio, potrebbe quindi essere realizzata inserendo in FF00 il valore binario 00001110, cioè 14 in decimale e $0E in esadecimale.

Per non restare solo in campo teorico, resettate il computer e caricate il listato 5 (disessamblato 1). Cancellate la riga 20 e date il Run, poi entrate in Monitor. Togliamo la parte che modifica l'interrupt (per qesto esempio non è indispensabile) digitando F 1300 1307 0, poi inseriamo le tre istruzioni

```
A 1308 LDA #$3C
A 130A STA $FF00
--------------------
A 131E RTS
```

A questo punto, per non creare equivoci su cosa si leggerà, riempiamo con un valore qualsiasi le locazioni di bank 0 equivalenti a quelle di bank 15 dove si trova il logo:

```
F 041BF 05000 EA
```
(carattere "P")

Usciamo dal Monitor ("X" e tasto Return), diamo NEW e digitiamo queste due righe:

```
10 Bank 0: Sys Dec("130d"): Stop
20 Bank 0: Sys Dec("1308")
```

Dando il Run al programma, con riga 10 le locazioni da $41BE verranno lette da bank 0, dove ci saranno le "P"; subito dopo il Cont, la riga 20 preleverà la scritta dal nuovo banco creato, e il comando Bank 0 (di per sè inutile) ci assicura di non trovarci nel banco 15. La nuova conformazione sarà data dal valore $3C inserito nel registro FF00, valore che in binario corrisponde a 00.11.11.00; avremo quindi un banco con la low Rom del Basic da $4000 a $8000, mentre nella memoria restante sarà presente solo Ram 0.

Concludendo

Siamo giunti al termine, e non ci resta che una breve riflessione. Per quanto lo si sfrutti al meglio, il nostro computer non può certo reggere il confronto con le prestazioni di un modello basato su un processore a 16 bit; considerando però tutto quello che si è detto (e quello che non si è detto) non si può non ammirare come sia stato "strizzato" bene nelle sue possibilità il pur vetusto (concettualmente, si intende) 8502. Ma lasciamo perdere le astrazioni, ora tocca a voi dare una strizzatina... ai vostri cervelli.

(List a pag. I)

LE AVVENTURE DI AMIGA

di Sandro Certi e Franco Toldi

Sfogliando gli ultimi numeri dell'autorevole 'Amiga World', forse la migliore tra le riviste americane dedicate a questa macchina, abbiamo notato come una larga parte delle recensioni che vi compaiono riguardino i giochi d'avventura.

Avrebbe potuto trattarsi di una scelta della rivista. Ci è bastato tuttavia esaminare i cataloghi degli importatori italiani di prodotti per Amiga per rilevare come le adventures vi facciano la parte del leone.

Questi elementi ci hanno fatto innanzitutto pensare che il genere gode di un eccellente stato di salute. Non a caso infatti, in presenza di una nuova macchina, le softerhouses più importanti hanno investito in così larga misura sulle avventure.

Ci è sembrato poi che la produzione di adventure games per Amiga potesse costituire un punto di osservazione privilegiato per rilevare i mutamenti che stanno avvenendo nell'intero genere. A tale fine abbiamo così esaminato una larga parte dei prodotti presenti sul mercato.

Non ci siamo trovati di fronte ad un sottogenere egemone con a lato delle varianti di scarso rilievo. Le avventure esaminate si distribuiscono invece in alcune categorie ben differenziate tra loro. Ciascuna di queste categorie ci sembra aspiri a divenire un sottogenere ben preciso, quasi a fissare uno standard per la futura produzione.

In questo articolo esamineremo due di questi filoni a nostro avviso particolarmente degni di attenzione: la produzione Cinemaware, dai cui sviluppi molto ci attendiamo, e le avventure 'wimp', facenti largo ricorso a grafica interattiva, che forse preludono all'affermarsi di un nuovo standard.

Perchè i lettori possano seguire meglio quanto detto, ci proponiamo di accompagnarli nell'esame fatto. Da un lato questo ci consentirà di delineare meglio le caratteristiche delle due tendenze, dall'altro permetterà al lettore di coglierle nello specifico di giochi ritenuti esemplificativi.

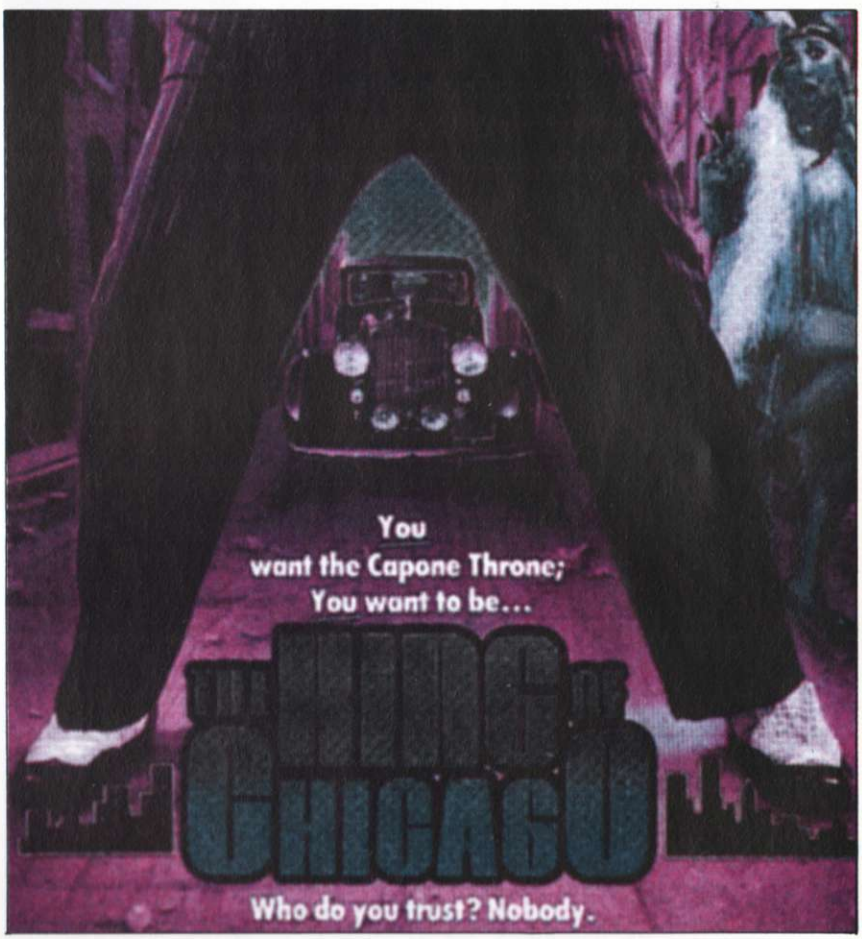
Cinemaware e wimp: sono queste le nuove parole magiche dei maghi degli anni '90 PC ti aiuta a scoprirne la formula

Cinemaware

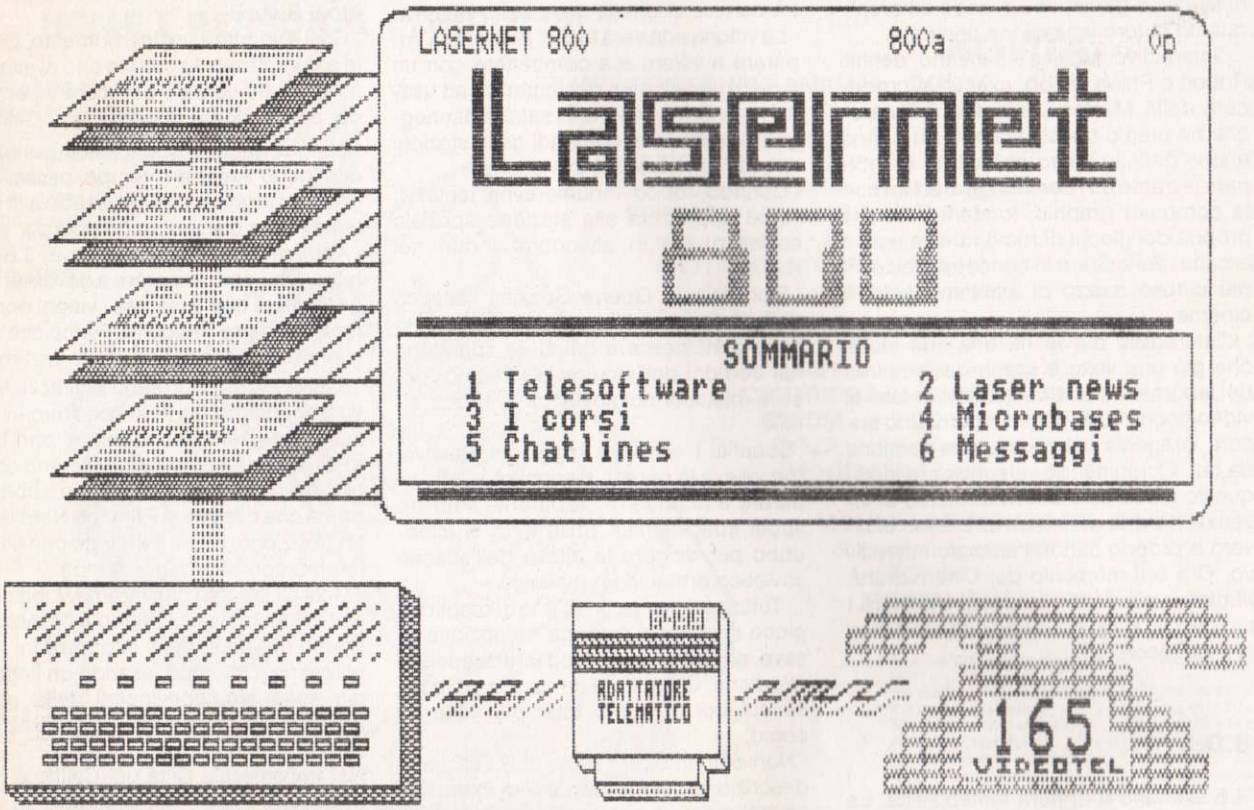
"In a word, Sinbad and S.D.I. are the prime cuts of today's adventure menu". Così Gary Fields in un entusistico articolo sul numero di ottobre di Commodore Magazine.

Defender of the Crown, S.D.I., Sinbad and the throne of the Falcon, e l'attesissimo The King of Chicago, sono i primi prodotti di una nuova linea di giochi che la Mindscape ha voluto chiamare Cinemaware.

E' proprio in questa parola, Cinemaware, che a nostro avviso va cercato quel



UN'EMOZIONE DA 1200 BIT AL SECONDO



- La potenza di una banca dati, la dinamica di un quotidiano.
- L'unico servizio telematico italiano con le notizie in tempo reale sul mondo dell'informatica.
- Il solo accessibile tramite la rete nazionale Videotel presente in piu' di 32 distretti telefonici (oltre 1000 comuni!).
- Con LASERNET 800 potrai caricare programmi in TELESOFTWARE, chiacchierare in diretta con tutta Italia sulle CHATLINES, editare un tuo spazio personale su PRIMA PAGINA, leggere le notizie piu' interessanti di LASER NEWS e migliorare la tua programmazione con i nostri corsi.
- Oltre 5000 pagine consultabili 24 ore su 24.
- Il nostro servizio ti costa ogni giorno meno della meta' di un quotidiano!

Per avere maggiori informazioni sul servizio compila il tagliando e spediscilo a:
LASERNET 800
VIA G. MODENA, 9
20129 MILANO - T.02/200201

PROVALA!

Desidero ricevere maggiori informazioni su LASERNET 800

Cognome..... Nome.....
Via.....
Citta'.....Prov....
CAP..... TEL...../.....
Data di nascita .../.../...

Il mio computer e' un:
Commodore ☐64 ☐128 ☐Amiga
☐MSX ☐BBC ☐Atari ST ☐PC
Spectrum ☐48K ☐Plus ☐128
☐ Ho gia' un adattatore telematico

Pc

qualcosa che rende questi prodotti diversi dagli altro giochi di avventura. Non è tanto la presenza di momenti di arcade, o di strategic game, che li rende differenti, quanto la loro ispirazione generale.

'Interactive Movies', li hanno definiti Robert e Phillis Jacob, executive producers della Master Designers Software che ha creato questi giochi. Essi hanno anche definito il loro obbiettivo: combinare le trame ed i personaggi dei film con la computer graphic, fondere modalità proprie dei giochi di ruolo, di strategia e arcade, per entrare in concorrenza con il più diffuso mezzo di intrattenimento, il cinema.

L'interactive movie ha una sua storia che già una volta è scesa dall'empireo dei laboratori del M.I.T. per mischiarsi ai videogiochi. Tutti, penso, ricordiamo ancora Dragon's Lear, nella sua versione da bar. Combinando videodischi e laser, questo gioco permetteva all'utente di vivere la propria avventura all'interno di un vero e proprio cartone animato interattivo. Ora è il momento del Cinemaware. Vediamo più da vicino due prodotti tra i più riusciti: S.D.I e Sinbad and the Trone of the Falcon.

S.D.I.

Lo scenario è di pura fantapolitica. La Strategic Defense Initiative, meglio conosciuta dai mass media come 'Guerre Stellari, è una realtà. Ne è seguito immediatamente un contraccolpo in Unione Sovietica, i cui dirigenti ritengono che essa sarà impiegata non a scopi difensivi, ma per portare un attacco termonucleare al loro paese, privo di possibilità di risposta.

In U.R.S.S. scoppia così un colpo di stato. Il K.G.B. occupa le postazioni missilistiche sovietiche e lancia un attacco ai satelliti del sistema S.D.I. Una volta che questi saranno distrutti, sarà così possibile lanciare un attacco termonucleare verso gli U.S.A.

A Mosca tuttavia i moderati chiedono l'aiuto degli U.S.A. per spegnere la rivolta e mettono a loro disposizione la propria stazione spaziale. Questa è comandata dall'eroina della storia, Natalya Kazarian, abile ufficiale e donna di grande attrazione (il ricordo di 'Dalla Russia con amore' non è molto lontano).

Un uomo solo può salvare milioni di americani ed il mondo libero: il capitano Sloan McCormick. E' nel suo ruolo che il giocatore dovrà calarsi.

Molti compiti lo attendono prima di poter tirare un momento il fiato. Innanzitutto dovrà abbattere una flotta di caccia spaziali del K.G.B., poi impiegare la difesa laser del S.D.I. per distruggere le diverse ondate di missili sovietici. Solo allora potrà correre in difesa della bella Natalya.

La vittoria non sarà facile. Bisognerà imparare a volare e a combattere con un super spacefighter, poi imparare ad usare i laser S.D.I., riparare i satelliti danneggiati e mettersi ai comandi delle stazioni spaziali americane e sovietiche.

Quando, dopo innumerevoli tentativi, l'eroe approderà alla stazione spaziale sovietica, qui lo attendono i duri del K.G.B.

Ricordate in Guerre Spaziali l'attacco delle truppe dell'Impero, il rapimento della principessa e quindi le sparatorie nei corridoi dell'astronave? L'atmosfera sulla piattaforma sovietica è molto simile.

Sconfitti i nemici e raggiunta Natalya, (ne valeva la pena!), bisognerà subito riparare e rifornire di carburante il nostro super spacefighter. Sulla terra ci attendono per dirigere la difesa dall'attacco sovietico ormai in imminente.

Tutto così, senza un attimo di respiro. Il gioco infatti non prevede un'opzione di save, sembra per non togliere suspence alla storia. C'è tuttavia un'opzione di pause cui noi abbiamo fatto più volte ricorso.

Non entriamo, nel caso di S.D.I. nella descrizione dei meccanismi di interazione del gioco. Lo faremo meglio in Sinbad, data la loro maggiore complessità. Per S.D.I. rileviamo solo che sono arcade-style e richiedono al giocatore buoni riflessi e una grossa perseveranza.

Sinbad and the Trone of the Falcon

Molto, molto lontano nel tempo e nello spazio da S.D.I. sorge il mondo di Sinbad. In esso confluiscono certo l'atmosfera delle Mille e una notte, ma soprattutto quella che emanda dalle molte versioni cinematografiche che la storia di Sinbad ha avuto.

La situazione in cui siamo proiettati è molto critica. Alla corte del Califfo è aperta la lotta per la successione. La vita dell'erede Harun è in pericolo e così quella di sua sorella Sylphani. Le armate del Principe Nero, lui pure figlio del Califfo, sono in marcia sulla capitale. Le forze del Califfo sono in allarme ma mancano ordini precisi.

Un malefico incantesimo ha poi reso incandescente il tutto: il Califfo è stato notte tempo tramutato in un falcone. Se non sarà trovato per tempo un rimedio il sortilegio sarà irreversibile. Harun non potrà essere procamato Califfo e le armate del Principe l'avranno vinta.

Ma Sylphani può contare sull'aiuto di un eroe, Sinbad.

Così è venuto il nostro momento. Difficile e complesso il compito che ci attende. Innanzitutto cercare per mare e per terra chi possa recare guarigione al Califfo.

Lunghi viaggi e sconosciuti pericoli ci attendono mentre il tempo passa. Una grande clessidra con della sabbia in continua, lenta discesa ce lo ricorda.

Abbiamo una splendida nave, il Sabaralus. Dovremo imparare a governarla ed a viaggiare con essa. Ma i viaggi non saranno sempre tranquilli. Pericolose scogliere ci attendono: sapremo uscirne?

A terra le cose non sono più facili. Molte volte incontremo il Principe Nero in persona e dovremo confrontarci con lui in duelli arcade-style. Poi gli Pteranoxis, feroci esseri alati, che dovremo abbattere prima che rivelino al Principe Nero la nostra posizione. Infine il Ciclope che affronteremo con una fragile fionda.

Molti gli incontri che faremo. Libitinia, Iris la zingara, lo Sciamano e il Genio. Ugnuno possiede delle informazioni a noi utilissime. Dovremo, usando un linguaggio adeguato, convincerli della nostra causa ed ottenere la loro collaborazione. Non sarà facile!

Nel frattempo la Città del Califfo è cinta d'assedio. Siamo noi a dirigere la resistenza. Guai se le difese cadessero.

Per compiere la nostra opera abbiamo a disposizione un vero e proprio sistema comunicativo. Esaminiamolo.

Cominciamo con i dialoghi. Essi avvengono tramite menu a scelta. I puristi forse rimpiangeranno l'uso della tastiera, ma mano mano che si addentreranno nel gioco scopriranno di quanta ricchezza sono dotati. Chi ha già giocato 'Law of West' resterà meravigliato di fronte alla complessità dell'interazione qui possibile.

Le scene di lotta. Qui è il joistck a farla da padrone, affiancato dal mouse. Abbiamo trovato questi momenti ben inseriti nel contesto della storia e ampiamente godibili.

L'assedio. Siamo di fronte ad un classico gioco di strategia. Non si deve pensare ad un qualcosa di staccato dalla storia. Proprio mentre noi viaggiamo per isole lontane, le forze nemiche attaccano e, se non sapremo opporre delle mosse valide, tutto sarà perduto. Dovremo quindi tornare di frequente a questa battaglia e intervenirvi. Battaglia e viaggi procedono infatti in simultanea. Noi diamo degli ordini e partiamo. Nel frattempo le armate marciano e se non interveniamo per tempo...

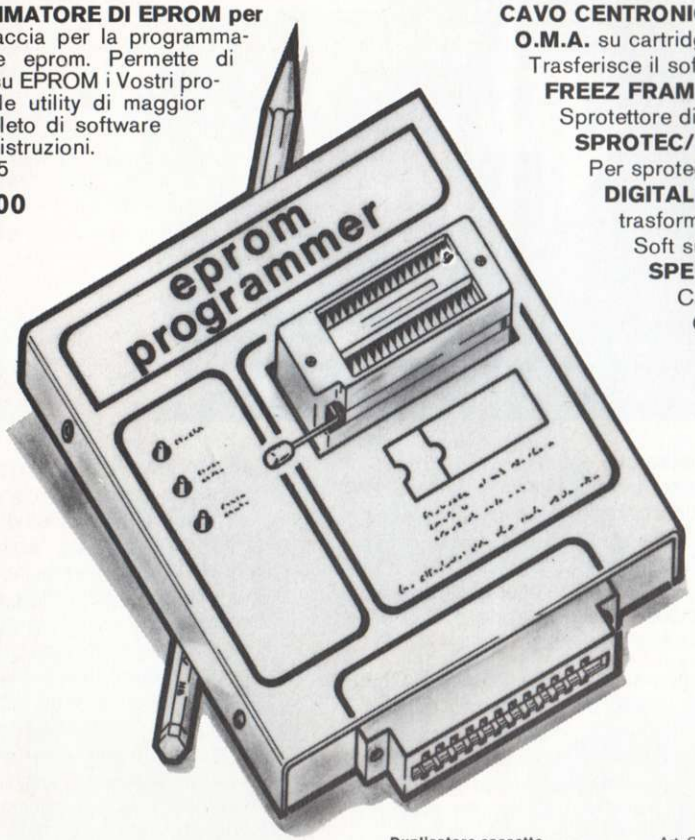
computer service

ACCESSORI PER COMPUTER HOME E PERSONAL COMPUTER

VENDITA PER CORRISPONDENZA

NOVITÀ

PROGRAMMATORE DI EPROM per c64. Interfaccia per la programmazione delle eprom. Permette di archiviare su EPROM i Vostri programmi o le utility di maggior uso. Completo di software su disco e istruzioni.
Art. CD 925
L. 160.000



studio ADVER

CAVO CENTRONICS AMIGA Art. CD 112 **L. 38.000**

O.M.A. su cartridge per c64 Art. CD 130 **L. 60.000**
Trasferisce il soft protetto e TANTE altre utiliti.

FREEZ FRAME per c64 Art. CD 132 **L. 55.000**
Sprotettore di programmi su nastro e su disco

SPROTEC/64 (isepic) Art. CD 910 **L. 60.000**
Per sproteggere i programmi del c64

DIGITALIZZATORE AUDIO per c64 Art. CD 915 **L. 89.000**
trasforma le voci in segnali digitali.
Soft su disco.

SPEED CONTROLLER per c64 Art. CD 920 **L. 35.000**
Cartuccia per ottenere l'effetto moviola.

CARTRIDGE DI PROGRAMMAZIONE EPROM per il CD 925. Art. CD 930. **L. 50.000**
Evita il caricamento del soft dal disco.

MODEM 300 baud per c64 **L. 156.000**
Art. CD 905

MOUSE-LOGIMOUSE C7 - 3 TASTI **L. 275.000**
con software per PCXT
Art. PC 365

MODEM V21 V23 seriale **L. 360.000**
per PCXT Art. PC 375

Vaschetta floppy in plexiglass (x 90 pz. con chiave)	Art. CD 780	L.	37.000
Kit pulizia testine registratore	Art. CD 815	L.	13.500
Kit pulizia disk drive	Art. CD 820	L.	20.000
Kit pulizia video antistatico	Art. CD 825	L.	12.000
Kit pulizia tastiera	Art. CD 830	L.	16.500
Foratore disk in plastica	Art. CD 840	L.	10.000
Speed dos plus Kit Velocizza il floppy di circa 20 volte. Per c64	Art. CD 900	L.	68.000
Eprom 2764 utilizzabile con l'articolo CD 925	Art. CD 950	L.	8.000
Eprom 27128 utilizzabile con l'articolo CD 925	Art. CD 952	L.	12.000
Stabilizzatore elettronico di tensione 500 W Con filtri e protezioni.	Art. CD 160	L.	430.000
Adattatore joystick per c16	Art. CD 225	L.	10.500
Adattatore registratore per c16	Art. CD 226	L.	19.500
Nastro inchiostrato per MT80	Art. CD 610	L.	14.000
Nastro inchiostrato per Tally MT180	Art. CD 611	L.	16.500
Nastro inchiostrato per Tally 1000 e Honeywell	Art. CD 612	L.	9.500
Nastro inchiostrato per Commodore MPS 801	Art. CD 614	L.	13.000
Nastro inchiostrato per Commodore MPS 802	Art. CD 616	L.	15.000
Nastro inchiostrato per Commodore MPS 803	Art. CD 618	L.	18.000
Pacco carta lettura facilitata 24"x11" 500 fogli	Art. CD 630	L.	13.500
Supporto stampante in plexiglass "fume" normale	Art. CD 660	L.	45.000
Supporto stampante in plexiglass "fume" rinforzato	Art. CD 670	L.	57.000
Disk 5" Singola Faccia Doppia Densità - 10 pezzi	Art. CD 700	L.	25.000
Disk 5" Doppia Faccia Doppia Densità - 10 pezzi	Art. CD 702	L.	30.000
Disk 3" Singola Faccia Doppia Densità - 10 pezzi	Art. CD 703	L.	60.000
Nastri magnetici C10 digitali 10 pezzi	Art. CD 712	L.	20.000
Nastri magnetici C15 digitali 10 pezzi	Art. CD 714	L.	21.000

Duplicatore cassette Copia con un registratore normale. Per c64 c128 vic20	Art. CD 102	L.	30.000
Copiatore programmi Copia con due registrari commodore. Per c64 vic20 c128	Art. CD 103	L.	30.000
Interfaccia radio Collega la radio al computer. Per c64 c128 o vic20	Art. CD 104	L.	30.000
Kit allineamento registratori c64 c128 vic20 Kit con strumento indicatore, nastro e cacciavite.	Art. CD 105	L.	45.000
Alimentatore per c64 e vic20	Art. CD 106	L.	38.000
Batteria tampone con batterie ricaricabili - Alimenta il c64 o vic20 in assenza di corrente per 30'	Art. CD 107	L.	118.000
Commutatore antenna tv/computer	Art. CD 108	L.	9.500
Tasto reset per c64 vic20	Art. CD 109	L.	5.500
Turbo Dos Velocizza il drive di circa 6 volte. Per commodore 64	Art. CD 115	L.	35.000
Penna ottica grafica per c64 (soft su disco)	Art. CD 121	L.	39.000
Penna ottica grafica per c64 (soft su nastro)	Art. CD 125	L.	39.000
Cuffia per Commodore per vic20 c16 c64 c128	Art. CD 150	L.	19.000
Copritastiera in plexiglass per c64 c16 vic20	Art. CD 750	L.	13.000
Copritastiera in stoffa per c64 c16 vic20	Art. CD 760	L.	10.000
Vaschetta floppy in plexiglass (x 40 pz. con chiave)	Art. CD 770	L.	30.000

SCONTI AI SIGNORI RIVENDITORI
TUTTI I PREZZI SONO COMPRENSIVI DI IVA. NON SI ACCETTANO ORDINI INFERIORI A L. 30.000.

BUONO DI ORDINAZIONE

04187

NOME

COGNOME

INDIRIZZO N.

C.A.P. CITTÀ

PROVINCIA P. IVA e/o Cod. Fisc.

VOGLIATE INVIARMI IN CONTRASSEGNO

Qt.	Cod. Art.	L.
Qt.	Cod. Art.	L.
Qt.	Cod. Art.	L.
Qt.	Art.	L.
PAGHERÒ AL POSTINO PIÙ SPESE POSTALI.		L.

PER ORDINI TELEFONICI: 0522/661471-661647

computer service s.n.c.

Via B. Cellini, 4 - 42017 NOVELLARA (R.E.) tel. 0522/661647

Il tempo rappresentato dalla clessidra, è il vero elemento unificatore del gioco. Il suo lento ma implacabile trascorrere unifica le nostre azioni e quelle dei nostri nemici in una storia. La nostra.

Avventure 'Wimp' - Deja Vu

Il secondo gruppo di avventure esaminate ci sembra si proponga come riferimento l'ambiente wimp proprio di Amiga, un ambiente cioè in cui l'utente fa uso per la comunicazione di finestre, icone, mouse e puntatori. E' l'ambiente, per intenderci, tipico del Workbench. L'ambiente cioè che si appoggia sulle sottostanti routines di interfacciamento di Intuition per la propria interazione grafica.

Sono 'Tass time in Tonetown' e 'Mindshadow' della Activision, 'Uninvited' della Mindscape, 'Ghost' della Headbanger e 'Deja Vù, ancora della Mindscape.

Questi giochi in maggior o minor misura fanno ricorso a finestre, icone, mouse e puntatori e più di rado alla tastiera per organizzare la comunicazione tra giocatore e gioco.

Per esaminarne più da vicino il sistema conunicativo addentriamoci in uno di essi. Quello dove l'utilizzo dell'ambiente wimp è più spinto, Dejà Vu.

Due parole sulla storia. In una ambientazione da giallo hardboiled, il giocatore si risveglia, dolorante nella toilette di un bar. Non sa cosa gli sia accaduto nè chi egli sia. Solo risalendo una lunga scala di indizi abilmente sparsi e sfuggendo a vari nemici che cercheranno di ostacolare, talvolta con le armi in pugno, la sua ricerca, egli riuscirà alla fine a scoprire il mistero della sua identità e la trama in cui è coinvolto.

Veniamo ora al sistema comunicativo che il giocatore ha a sua disposizione e per prima cosa alle cinque 'finestrè in cui lo schermo è suddiviso. Esaminiamole una per una e poi nei loro rapporti reciproci.

La 'grafic window'. Posta sul lato sinistro dello schermo, ne occupa un quarto circa. La grafica è buona e - cosa più importante - i disegni in essa contenuti sono interattivi, possono cioè essere oggetto di lavoro da parte del giocatore.

In primo luogo si potrà, portando il puntatore del mouse sui particolari del disegno stesso, telefoni, abiti, mobili o altro, provocare l'apertura di finestre contenenti informazioni su tali oggetti.

Congiungendo poi quanto compare nella 'grafic window' con i comandi Examine, Open, Close, Operate, Go e Hit, esposti nella 'Command Window' - un'altra finestra che esamineremo subito - si provocheranno le azioni desiderate. Basterà cosi portare il puntatore del mouse su Examine, premere il pulsante sinistro e ripetere le stesse operazioni su un oggetto per ottenerne la descrizione. Fare la stessa cosa con una porta e ottenerne l'apertura. E così via.

La Command Window. Posta sulla parte sinistra in alto dello schermo espone i comnadi di cui il giocatore può usufruire mediante mouse. Abbiamo visto prima come, in combinazione con la 'Grafic Window', costituisce uno dei principali mezzi di comunicazione tra giocatore e gioco.

La 'Inventory Window'. Mostra gli oggetti di cui il giocatore mano mano entra in possesso. Ciò avviene trasportando, mediante mouse, l'oggetto dallà Grafic Window' alla 'inventory Window' stessa. L'oggetto contenuto nella 'Inventory Window' può essere ulteriormente 'lavorato da parte del giocatore. Basterà ad esempio portare il puntatore su 'Open', premere il pulsante sinistro e ripetere la stessa operazione su un oggetto contenuto nell'Inventory Window per provocare l'apertura di una nuova finestra esibente il suo contenuto. Sottoponendo ad esempio a questo trattamento la pistola che compare in una delle prime scene, veniamo così sapere che tre dei colpi contenuti nel caricatore sono stati esplosi.

La Text Window. Costituisce la parte testuale dell'avventura. Può contenere sino a sei righe di testo. E' possibile rivedere in questa finestra il testo precedentemente scorso. Basta operare con il mouse sulla barra di scorrimento posta sul lato destro della finestra.

La Self Window. Per il gioco rappresenta il giocatore stesso. In congiunzione con i comandi Open e Close della 'Command Window' provoca l'apertura o la chiusura della 'Inventory Window'.

La Exit Window. Evidenzia al giocatore, mediante piccoli quadrati, le vie di uscite dal luogo in cui egli si trova. E' naturalmente manovrabile da mouse.

Ci siamo dilungati sulle modalità comunicative di questo gioco onde mettere in luce l'interessante uso che in esso viene fatto della parte grafica. Come da tempo attendevamo, dopo giochi con grafica seducente ma tutto sommato solo illustrativa, in questo caso siamo di fronte finalmente ad un uso interattivo della grafica stessa.

Se da un lato sottolineiamo quindi l'interesse del gioco per la sua trama e la sua ricchezza, dall'altro ne evidenziamo la componente comunicativa.

Non pensiamo di sbagliare prevedendo che Deja Vu costituirà un punto di riferimento per la futura produzione di giochi d'avventura. I recentissimi Uninvited della Mindscape e Ghost! della Headbanger ci sembra lo confermino.

Si conclude così il nostro breve viaggio nei due più interessanti sottogeneri, forse dovremmo chiamarli standard, che hanno fatto la loro comparsa nel variegato mondo dei giochi d'avventura. Non possiamo che essere ottimistici sul loro futuro. Da loro ci attendiamo molto.

Le avventure di Amiga non si esauriscono qui. C'è dell'altro... Giochi come The Pown, The Guild of Thives della Magnetic Scrolls da un lato o come Barbarian della Psygnosis non possono essere ignorati. Ma, come dice un film di successo, questa è un'altra storia.

KLEENTECH

Il computer da la sicurezza dei dati Kleentech la nitidezza

Polvere, umidità, fumo alterano le caratteristiche di nitidezza del video rendendo difficoltosa la consultazione dei dati visualizzati. Per questo è importante mantenere in perfette condizioni di pulizia e leggibilità schermo e tastiera. Da oggi esiste un prodotto specifico ed estremamente funzionale: si chiama Kleentech. Kleentech è un fazzolettino di uno speciale tessuto umidificato con un solvente che non striscia lo schermo, non lascia alcun residuo, non danneggia nè corrode la superficie, ma minimizza l'accumulo di polvere perchè antistatico. Utile, pratico, sicuro, Kleentech è sempre a portata di mano nella sua confezione singola.

Importatore esclusivo:
I.B.C.
Viale Corsica, 72 - Milano
Tel. 02/7387398-744698

ibc INTERNATIONAL BUSINESS CONSULTANTS

SUPER WINDOW

di Carlo Notarianni

Il QL offre, si sà, la possibilità di aprire più finestre video, ma purtroppo mancano dei comandi per gestire finestre "reali"; ovvero, posso aprire quanti canali voglio ma non ho a disposizione un comando che mi rinfreschi il contenuto di una finestra andato perso dalla scrittura in un altro canale.

Da questa premessa è nata l'idea di scrivere un programma che aggiungesse nuovi comandi per salvare e rinfrescare sul video una qualsiasi finestra.

Il sistema utilizzato non necessita delle risorse speciali del sistema operativo ma, per motivi di velocità, si deve utilizzare il linguaggio macchina (ho utilizzato l'ottimo ASSEMBLER WORKBENCH della TALENT).

Funzionamento

Il display-file inizia alla locazione 131072 (decimale) ora, data la finestra **LARG, ALT, XPOS, YPOS**, il suo punto di inizio nel display-file è dato dalla formula seguente:

INIZ=131072+XPOS/4+128*YPOS.

La riga di ogni finestra richiede **LARG/4** Bytes di memoria video e il numero delle righe è dato dal secondo parametro (**Alt**); una finestra dunque occuperà **LARG/4*Alt** Bytes di memoria video, inoltre il punto di inizio di ogni riga è distante dal punto di inizio della riga succesiva 128 Bytes.

Sapendo ciò è facile salvare una finestra in RAM: si copiano i bytes di una finestra uno dopo l'altro in un buffer che ricopieremo nel display-file quando vorremo ripristinare la finestra salvata.

Useremo inoltre un HEADER iniziale che ci permetterà di ricopiare la finestra salvata non solo nel punto in cui appariva, ma anche in tutte le posizioni possibili per le sue dimensioni.

Inoltre è possibile effettuare il refreshing della finestra salvata con tre tipi di over (-1,0,1 default over 0) per aumentare la gamma degli effetti possibili.

Sintassi dei comandi

Abbiamo due PROCedure ed una FuNzione che sono nell'ordine:

SCR__STORE 〈Larg,Alt,Xpos,Ypos, Buffer〉 (1' Sintassi)

SCR__STORE 〈[#n,]Buffer〉 (2' Sintassi)

SCR__REFRESH 〈Buffer [,Over][,Xpos,Ypos]〉

SCR__SIZE 〈Larg,Alt[,Xpos,Ypos]〉 (1' Sintassi)

SCR__SIZE 〈[#n]〉 (2' Sintassi)

Le parentesi angolari 〈 〉 indicano uno o più parametri obbligatori mentre le parentesi quadre [ ] indicano uno o più parametri opzionali.

Buffer è l'idirizzo di memoria, riservato con **RESPR** o con **ALCHP**, sufficente a contenere i bytes della finestra da salvare (per sapere quanti ne occorrono si usa **SCR__SIZE** come spiegato più avanti).

SCR__STORE come vedete, può avere due sintassi:

- la prima richiede 5 parametri obbligatori (le dimensioni della finestra, la sua posizione nello schermo e l'indirizzo buffer-RAM), se i parametri sono tutti O.K. la finestra indicata viene memorizzata a partire da Buffer+8 (dopo l'header dei parametri) altrimenti la procedura torna al basic con BAD PARAMETER o OUT OF RANGE aseconda dei casi;
- la seconda sintassi prevede uno o due parametri (numero di canale e Buffer) il primo parametro deve essere preceduto da # (cancelletto) ed indica il numero di canale associato allo schermo (DEVICE CON- o SCR-) che si vuole salvare (è previsto il default per il canale 1); se il canale è aperto ed associato allo schermo viene salvato in RAM a partire dall'indirizzo dato da Buffer+8 (Buffer=secondo parametro) altrimenti viene dato un errore opportuno.

SCR__REFRESH invece richiede da 1 (obbligatorio) a 4 parametri vediamo degli esempi:

SCR-STORE 448,200,32,16 TO Buffer

Viene salvata in **Buffer** la finestra indicata.

SCR__STORE #2,Buffer

Viene salvata in **Buffer** la finestra del canale 2.

SCR__STORE Buffer

Viene salvata in **Buffer** la finestra del canale 1.

SCR__REFRESH Buffer

Viene ripristinata (nello stesso punto da dove era stata salvata) con OVER 0 la finestra precedentemente salvata in Buffer.

SCR__REFRESH Buffer,Ov

Come sopra ma usando OVER Ov.

SCR__REFRESH Buffer,Xpos,Ypos

Viene ripristinata, a partire dalle coordinate Xpos,Ypos, la finestra precedentemente salvata in Buffer.

SCR-REFRESH Buffer,Ov,Xpos,Ypos

come sopra con OVER Ov.

SCR__SIZE, come vedete ha due sintassi possibili:

1' sintassi (senza numero di canale):

A=SCR__SIZE (Larg,Alt)

ritorna in A il numero di bytes necessari per salvare una finestra delle dimensioni indicate.

A=SCR__SIZE (Larg,Alt,Xpos,Ypos)

Come sopra ma controlla se Xpos e Ypos sono nel range adeguato.

2' sintassi (con numero di canale)

A=SCR_SIZE (#n) (default #1)

Ritorna in **A** il numero di bytes necessari per salvare lo schermo video del canale #N (controlla se il canale è aperto e se associato al video altrimenti torna con l'errore opportuno).

A=SCR_SIZE

Ritorna in **A** il numero di bytes necessari per salvare lo schermo video del canale 1.

Il programmino 'ANIMATE-BAS' mostra un esempio d'uso delle routine e per la delizia dei QL-USERS il programmino mostra un esempio di animazione in tempo reale ottenuta ruotando una figura di PI/n gradi N volte.

Il programma inizia caricando da microdrive (o floppy) i codici precedentemente salvati ed attiva i nuovi comandi; quindi disegna una serie di figure che memorizza in un buffer al termine viene mostrata sul video la figura in movimento.

Ora se premete i tasti freccia (associati ai tasti indicati per uno step maggiore) potrete spostare il riquadro (la finestra) dove la figura appare ad esempio in un angolo, lasciati i tasti freccia la figura apparirà di nuovo sul video e ricomincerà il movimento in tempo reale.

Nel listato basic del programma ANIMATE_BAS sono presenti tre procedure per salvare una finestra (WINDOW_SAVE) o un canale video (CHANNEL_SAVE) sul supporto indicato (microdrive o floppy) in DEV_name$ (DEVice & Filename); WINDOW_LOAD invece riporta sul video una finestra (o canale) precedentemente salvati con WINDOW_SAVE o CHANNEL_SAVE; essi rappresentano un valido esempio di utilizzo dei nuovi comandi presentati; per quanto riguarda i comandi (o funzioni) **FLEN**, **ALCHP**, **RECHP** potete usare sia quelli presenti sul TOOLKIT di TEBBY che le analoghe estensioni apparse su *Personal Computer*.

Per avere una copia operativa dei codici dovete battere il listato del programma caricatore GRAF_BAS il quale se non trova errori (eventualmente corregeteli) salva in MDV1-GRAF-CODE il codice generato.

Per attivare i nuovi comandi date:

```
10 GRAF=RESPR(900)
20 LBYTES MDVn-GRAF-CODE, GRAF
30 CALL GRAF
40 ..... IL VS PROGRAMMA ....
```

L'header iniziale del buffer contenente la finestra memorizzata, contiene all'inizio 4 WORD (2 bytes ciascuna) nelle quali sono trascritte rispettivamente l'**AL**tezza, la **Larg**hezza la **Xpos** (posizione x nello schermo) e la **Ypos** (posizione y della finestra nello schermo); dando quindi il comando:

FOR I=Buffer to Buffer+8 step 2: PRINT PEEK-W(I) potrete leggere (o scrivere con **POKE_W**) le dimensioni della finestra salvata in **Buffer**.

Consiglio comunque di modificare eventualmente solo la posizione della finestra nello schermo (**XPOS** e **YPOS** terzo e quarto parametro) o, al limite, diminuire solo l'**ALT**ezza della finistra (secondo parametro) altrimenti la figura non verrà ripristinata regolarmente (i dati sono memorizzati sequenzialmente !!); in ogni caso, se passate parametri errati (fuori dai limiti consentiti) dando SCR_REFRESH otterrete l'errore OUT OF RANGE infatti, le routines controllano sempre e comunque che i parametri loro passati siano accettabili e nel range loro consentito.

I limiti per i parametri sono:

```
2<=LARG<=512
2<=ALT<=256
0<=XPOS<=(512-LARG)
0<=YPOS<=(256-ALT)
```

Per finire, ricordate che, se usati bene, questi comandi saranno per voi una preziosissima fonte di aiuto quanto vorrete gestire menù a discesa o fare degli input in programmi grafici o per operazioni di TAGLIA & INCOLLA e per tutto ciò che la Vostra inesauribile fantasia può partorire.

(List a pag. IV)

TANTI BUONI MOTIVI PER ABBONARSI A

VR VIDEOREGISTRARE

12 NUMERI AL PREZZO DI 10
solo 45.000 lire
invece
di 54.000 lire

•

PREZZO BLOCCATO
per tutta la durata dell'abbonamento

•

SICUREZZA
di non perdere neanche un momento

•

COMODITÀ
di ricevere la propria rivista preferita a casa

•

COSA STATE ASPETTANDO?

GARE CON L'ARCO

di Fabio Cavicchio

Questo gioco consiste nello scagliare 6 frecce contro un bersaglio mobile visto di lato. Il risultato della prova, visualizzato su un secondo bersaglio, questa volta visto frontalmente, dipenderà ovviamente dalla precisione del tiro ma anche dall'angolazione dell'arco; tale angolazione dipende dal tempo per cui si tiene premuto il tasto SPACE che è il tasto con cui si scagliano le frecce.

La freccia si conficca nel bersaglio solo se lo colpisce con una angolazione compresa tra 10 e 20 gradi, essendo quella ottimale di 15 gradi.

Ciò corrisponde ad un tempo di pressione del tasto SPACE da mezzo ad un secondo circa.

Tutti i movimenti sono stati ottenuti con un sistema di finestre che vengono fatte scrollare tramite i vari comandi del SuperBasic del QL.

La finestra #5 (vedi Figura 1) tramite l'istruzione **SCROLL** fa muovere il bersaglio; la #1 con il comando **PAN** fa scrollare la freccia e la #7 fa continuare il percorso della freccia nel caso in cui il bersaglio venga totalmente mancato.

Il punteggio finale è quello desumibile dalla somma dei sei punteggi parziali. Nel caso fosse stato raggiunto un nuovo record questo verrà aggiornato.

Durante il tracciamento dello schermo di gioco, ho ritenuto opportuno "spegnere" lo schermo per ottenere un risultato esteticamente migliore.

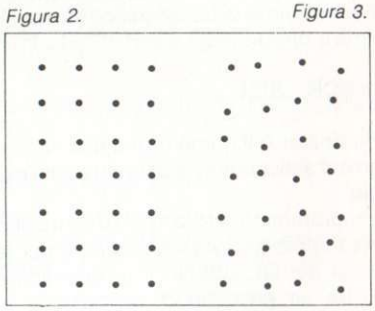
Per quanto riguarda l'"erba" essa è stata ottenuta con delle **POINT** sistemate in modo pseudo-casuale (per evitare che fossero stampati all'interno di un'altra finestra); sono stati plottati come se disposti a griglia ma aggiungendo un piccolo spostamento casuale (più o meno 2) che li fa apparire irregolari (vedi Figure 2 e 3).

Buon divertimento!

(List a pag. XII)

Figura 2.

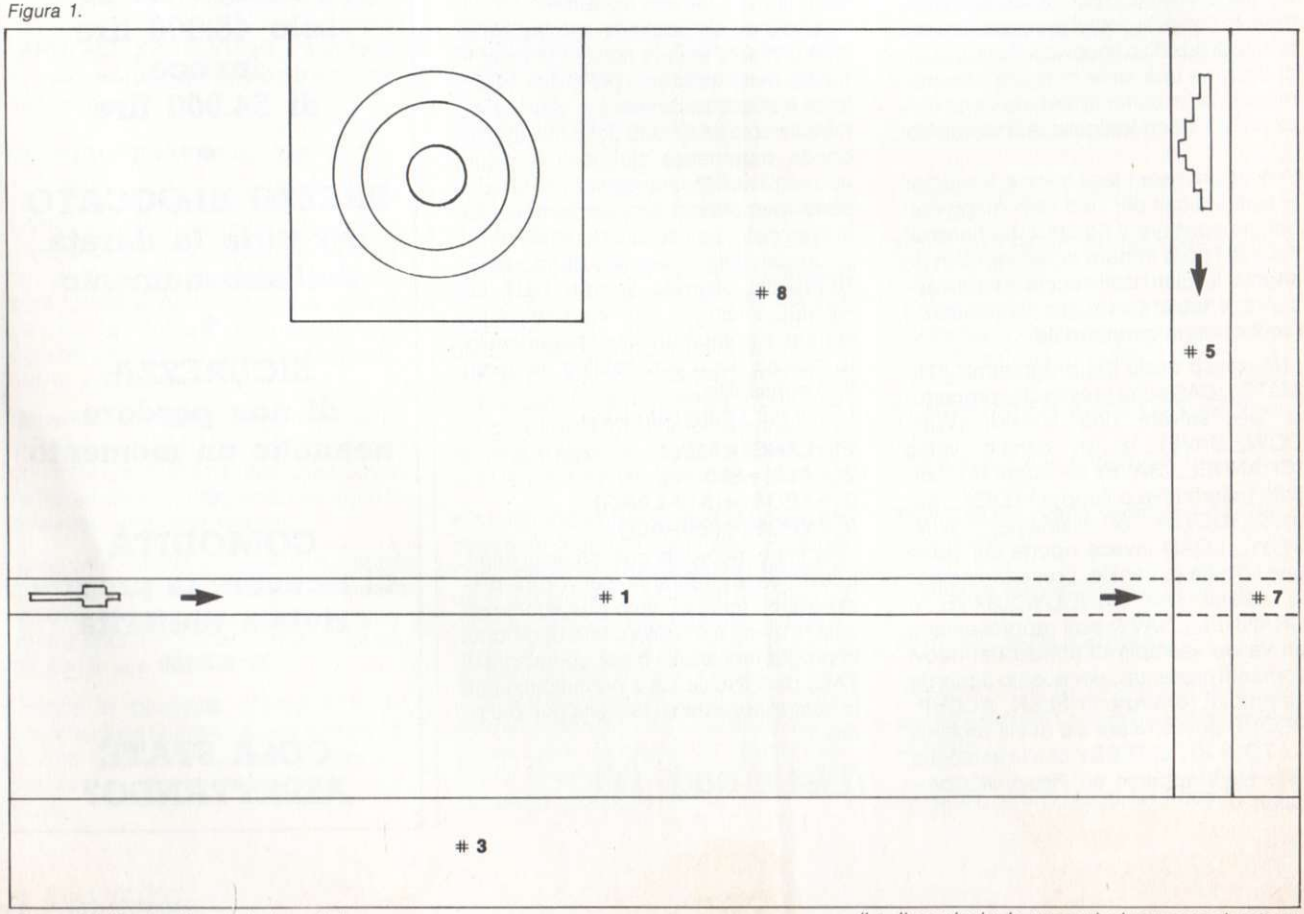
Figura 3.



Procedura "ERBA"

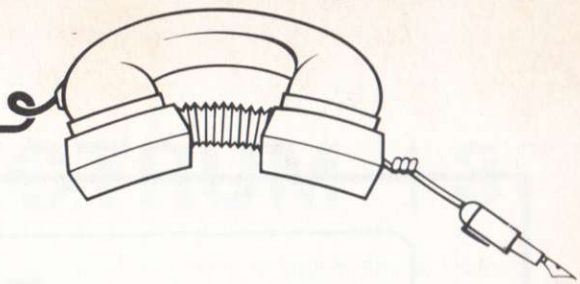
```
X=X + (RND-2 TO 2)
Y=Y + (RND-2 TO 2)
```

Figura 1.



Sistemi di finestre durante la fase di gioco.

(Le dimensioni e le proporzioni non sono rispettate)



(1) Non sarebbe possibile, una volta all'anno, che voi commercializzaste una cassetta con tutti i programmi pubblicati per lo Spectrum nel corso dell'annata ? (2) Ho acquistato l'interfaccia Disciple per lo Spectrum: è fantastica, ma non è compatibile con le stampanti Sinclair-dedicate (come la mia Alphacom 32). (3) Esiste un simulatore di Ms-Dos per lo Spectrum ? (4) A quando i programmi per il 128K+2 ? (5) Ho trovato un articolo che descrive un circuito che amplia la memoria accessibile da parte del microprocessore Z80, che ve ne pare ?

(G.Manea-Piovene Rocchette/VC)

(1) L'idea è più che buona, e ci penseremo seriamente. (2) Non abbiamo provato l'interfaccia con simili stampanti, ma deve pensare che, forse, i suoi costruttori le hanno considerate superate e non hanno implementato il loro uso. (3) No, per i limiti che lo Spectrum ha finora avuto con la memoria di massa. Ora sarebbe possibile, e proprio con il Disciple in suo possesso, che consente di disporre di circa 760 K di memoria di massa accessibile in tempi brevissimi su floppy-disk. Se vuole mettersi all'opera... .

(4) Presto: non ne sono ancora arrivati (data:22/7/1987), per cui qualcuno ce lo stiamo facendo noi, come un grosso programma musicale.

(5) A occhio è certamente possibile, dallo schema che acclude e che non possiamo, però, pubblicare. Da che rivista lo ha preso ?

Perchè non pubblicate routines per poter cambiare nome ai files presenti sui Microdrives per Spectrum, o addirittura anche per cambiare nome alla cassetta stessa ? *(Franco Schinco-Torino).*

Beh, francamente perchè finora non ci era mai passato per la mente. E' una idea che mettiamo in "coda" alle altre da realizzare (sono tante, e noi pochi, come il tempo che abbiamo): nel frattempo, nessuno impedisce ai lettori la costruzione di un simile programma di utilità.

(1) Avete intenzione di pubblicare un seguito del progetto del programmatore di EPROM , magari capace di scrivere sulle 27128 ?
(2) Ho trovato in un programma assembler le istruzioni DDh 44h e DDh 4Dh, che non si trovano nell'elenco ufficiale delle istruzioni Z80. In realtà corrispondono a LD B,IX(high) e LD B,IX(low).
(3) Montando uno Z80B sullo Spectrum sarebbe possibile portare il clock a 7Mhz, senza pregiudicare il funzionamento del tutto ?

(Fabio Ferrero-Bologna).

(1) Ricordiamo anzitutto cosa si intende per EPROM: sono delle ROM particolari che conmservano la caratteristica di sola leggibilità delle stesse, ma che è possibile scrivere con apparecchiature particolari, chiemati appunto "programmatori" che cancellano il contenuto della EPROM tramite una apposita luce ultravioletta e provvedono alla memorizzazione del nuovo programma. Il relativo progetto hardware è stato pubblicato sui numeri 11, 12 e 13 di Sinclair Computer, e permetteva la programmazione di alcuni modelli di EPROM, più precisamente dei chip contrassegnati 2516 e 2532, di capacità rispettivamente di 2 e 4 Kbytes . Se l'autore o qualche lettore ci propone modifiche o progetti anche per le 27128 (dalla capacità di 16 Kbytes), niente in contrario alla pubblicazione.

(2) Infatti nell'elenco ufficiale non ci sono. Mistero.

(3) Crediamo di no, perchè il sistema è configurato per il clock del processore Z80A, e quindi pensiamo che si avrebbero problemi insormontabili di sincronizzazione (ad esempio per il video): ricordiamo che all'interno del programma della ROM ci sono dei cicli di ritardo ben precisi, basati sul periodo tipico dello Z80A, e che verrebbero accelerati con un clock più alto, che d'altronde potrebbe anche non essere tollerato dagli altri componenti elettronici del sistema (ULA, RAM, ..).

Sul n.14 della rivista la rubrica "mailbox" è stata impietosamente tagliata in fase di fotocomposizione, per un banale errore. Ce ne scusiamo con i lettori, e in particolare con A.Luce di Cassano Murge, la cui lettera è stata interrotta nelle risposta.

Pubblichiamo ora la parte mancante della posta del n.14, riprendendo proprio dalla lettera di A.Luce. Scusateci ancora.

(1) Quale assemblatore/disassemblatore è stato usato per gli articoli di "Speciale Spectrum Sprotezioni" ? Ne potete pubblicare il listato?
(2) Vorrei che pubblicaste due mappe: (A) gli indirizzi delle varie routines in linguaggio macchina, ben spiegate ad una ad una in italiano. (B) gli indirizzi di tutte le porte di ingresso ed uscita dello Spectrum con le relative spiegazioni. *(A.Luce-Cassano Murge/BA)*

(1) L'autore non ce lo ha detto, ma, tra quelli disponibili sul mercato, si equivalgono tutti. Abbiamo già pubblicato un disassemblatore potentissimo su Sinclair Computer n.12 (mar/apr 1985), e un disassemblatore/rilocatore su Personal computer n.07 (ott.1986).

(2-A) .... cioè la ROM completa dello Spectrum! Che ha richiesto un libro di 236 pagine (commentato in inglese, notoriamente più conciso dell' italiano..) per essere spiegata. Pubblicandone un paio di pagine a numero potremmo finire circa nel 1999. Meglio imparare l'inglese (che serve in ogni caso), e leggersi "The Spectrum ROM Disassembly" di Logan/O'Hara, pubblicato dalla Melbourne House Publishers, no ?

(2-B) Le porte di I/O dello Spectrum sono esattamente 65535: alcune di esse hanno un uso pratico, come la 44244 (trovata a caso!) che serve per l'input da Joystick (provate a vedere i vari valori che assume in corrispondenza delle varie posizioni del joystick stesso, con l'istruzione BASIC

10 PRINT IN 44244: GO TO 10), e altre non sono usate, comunque una simile mappatura non esiste e, crediamo, nessuno avrà mai il coraggio di farla perchè ci vorrebbe troppo tempo. Se qualche lettore ce la fa , avrà scritto una pagina importante nella storia SPECTRale, pagina/e che Personal Computer sarà lieto di pubblicare.

GRAPHIC EXPANDER 128

Systems

Aggiunge al tuo Commodore 128 ben 14 comandi Basic espressamente dedicati alla gestione della grafica su schermo a 80 colonne (640x200 punti).

E' possibile ottenere il software in questione (solo su dischetto) compilando il coupon a fondo pagina e indirizzandolo a:

Systems Editoriale
Viale Famagosta, 75
20142 Milano

Modalità di pagamento

Al coupon va accluso un assegno di Lire 27.000 (comprensivo delle spese di spedizione) intestato alla Systems Editoriale.

Vi prego di inviarmi il dischetto Graphic Expander 128.

Nome .....
Cognome .....
Indirizzo .....
Cap ..... **Città** .....

Accludo assegno di Lire 27.000 (comprensive di spese di spedizione).

Firma .....

Posseggo uno Spectrum, al quale ho abbinato un modem Prism VTX 5000, acquistato in Inghilterra al modico prezzo di 34 sterline (compresa la spedizione!); esso funziona perfettamente (sono felicemente collegato a Videotel), ma mi sembra che possa colloquiare solo alle velocità di 75 o 1200 baud. Infatti con le banche dati che colloquiano a 300 baud non riesco ad ottenere l'agganciamento, anche perchè il manuale in dotazione non dà certo risposte esaurienti. Aiuto! *(Scaglioni Romano-Modena).*

Appello: in casi come questo accludete almeno le fotocopie delle specifiche tecniche della macchina. Mica ce le sogniamo. Detto questo, ci sembra strano che il VTX5000 abbia solo due velocità di commutazione, e perlopiù così opposte. Non si tratterà per caso dello standard di comunicazione V23, piuttosto ? Se non conosciamo nè specifiche, nè standard di comunicazione, nè la macchina stessa (non le abbiamo mica provate tutte!) come facciamo a darle una risposta ? Ci riscriva, per favore, cercando di essere più completo possibile.

Vorrei chiarimenti sulla linea 0: come si costruisce, e come si elimina ? *(Francesco Violentano-Terranova/Cosenza).*

La linea 0, generalmente usata per messaggi di copyright o per contenere routines in linguaggio macchina, si costruisce ponendo a 0 i due bytes rappresentativi del numero di linea della prima linea di istruzioni del programma. La locazione di inizio del programma è data dalla variabile di sistema (capitolo 34 del manuale) indicata con il nome simbolico di PROG, e si può conoscere dando il comando LET pippo=PEEK 23635+ +256*PEEK 23636:PRINT pippo. Ora, "pippo" rappresenta la locazione del byte meno significativo del numero di linea della prima linea del programma, e "pippo + 1" la locazione del byte più significativo. Per far sì che una linea abbia numero 0 basta quindi dare POKE pippo,0 e POKE (pippo+1),0.

Per eliminarla, è sufficiente una POKE a "pippo" di un valore opportuno, e cioè diverso da ogni altro numero di linea del programma (per evitare confusioni allo Spectrum e, soprattutto,a voi).

LO SPECTRUM +3

a cura di Andrea Gallo

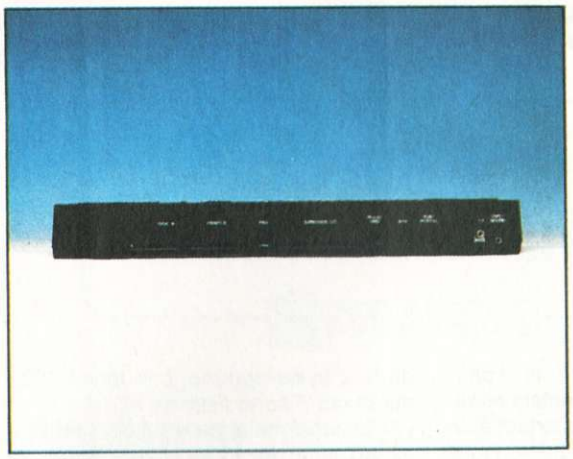
Sinclair ZX Spectrum 128K/+3 è il nome esatto del nuovo nato della Sinclair/Amstrad ed insieme al nome diviene ogni volta piò complesso e completo anche il computer.

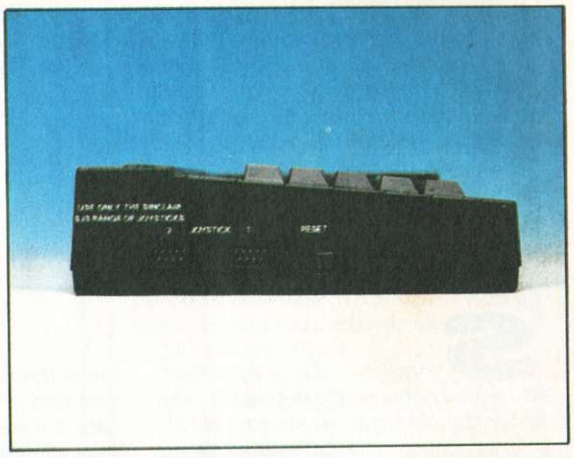
Il cambiamento piò vistoso, oltre al ritorno al colore nero, è sicuramente il "floppy disc drive" che ha preso il posto del vecchio registratore a cassette. Sul lato sinistro della "carrozzeria" rimangono invariate le due porte per joystick Sinclair SJS. Sul lato posteriore si notano invece diverse modificazioni: la presa audio è ora anche il socket per il registratore (il cui cavo di collegamento non è incluso nella confezione); il connettore del tastierino numerico è diventato una presa AUX priva di importanza; la presa MIDI è sempre presente insieme alla RS232C; la expansion port è mutata, essendo stata eliminata la linea a 9V percui le interfacce da essa alimentate inevitabilmente non funzioneranno piò (per prima l'interfaccia 1 con i microdrives, il cui copyright era rimasto alla Sinclair); la presa dell'alimentatore porta ben tre linee separate (una a +5V e due a + e - 12V); la presa d'aria sulla sinistra è stata sostituita da un doppio connettore: una porta Centronics e una porta per il drive esterno.

Il +3 (d'ora in poi lo abbrevieremo così) usa dischi Amstrad CF2 da 3" a doppia faccia alternabile: quando, cioè, avete esaurito un lato del disco, lo girate semplicemente e avete accesso ad altri 173K. Il disco infatti contiene 40 tracks di 9 settori da 512 bytes ognuno, che fanno 180K ma di cui 7 sono riservati al +3.

All'accensione appare sullo schermo lo stesso menu del +2 ad eccezione del fatto che il "tape loader" non attende piò files da nastro se non dopo aver appurato l'assenza di un disco nel drive: cerca infatti di caricare il file "*" in l/m oppure un loader basic registrato con il nome "DISK".

Per la prima volta uno Spectrum è anche CP/M compatibile, avendo un'opzione di sola RAM (eliminando temporaneamente le varie ROMs) e pare che la Locomotive Software, società autrice del +3 DOS, stia scrivendo un simulatore che permetta di caricare gli ottimi programmi della vasta biblioteca CP/M.





Il +3 Basic, fornito di nuove istruzioni, è purtroppo incompatibile con il +2, suo predecessore, in quanto non solo è cambiata la sintassi dei comandi per accedere alla RAMdisk ma la stessa è stata ridotta a 58K.

I nuovi comandi per l'uso dei dischi sono dunque in stile CP/M: prima del file name deve infatti essere inclusa una lettera, seguita dai due punti (:), che richiami il tipo di memoria di massa cui si vuole accedere: "A:" indica il drive incorporato nel sistema, "B:" il drive esterno opzionale, "M:" la RAMdisk e "T:" il registratore a cassette. In questo modo infatti non solo la sintassi è uguale per tutte le periferiche citate ma permette anche un facile movimento di files da nastro a disco: digitando infatti LOAD "T:" e poi SAVE "A:", ogni successivo LOAD attenderà files provenienti dal registratore a cassette ed ogni SAVE invierà i dati sul disco interno. Per i files noti come CODE di cui non si conosce nè l'indirizzo di partenza nè la lunghezza, è presente l'istruzione CAT "T:" che esegue la funzione di header reader. Il comando COPY nella sua nuova sintassi COPY "A:" TO "B:" esegue una copia settore per settore del disco nel drive A: sul disco nel drive B: e se avete un solo drive il +3 vi chiederà al momento opportuno di sostituire al disco sorgente il disco su cui volete eseguire la copia.

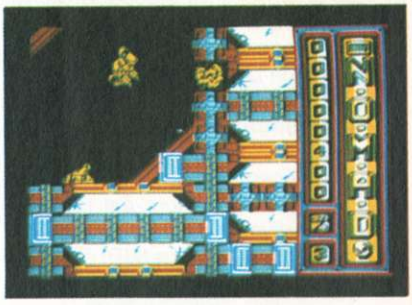
Le stampanti

Essendoci entrambe le porte RS232C e Centronics, ogni tipo di stampante funziona sul +3. L'istruzione LPRINT "testo" invia il "testo" sulla periferica selezionata dal comando FORMAT LPRINT "R" per la linea RS232Ç e FORMAT LPRINT "C" per la Centronics.

Gradita innovazione risulterà sicuramente l'istruzione FORMAT LPRINT "U", forma abbreviata per "Unexpanded", che permette di inviare caratteri di controllo alla stampante (è equivalente al canale "b" della vecchia Interfaccia 1).

FORMAT LPRINT "E" ripristina la situazione di default che corrisponde al canale "t" ovvero filtra i caratteri con codice ASC II inferiore a 32 ed espande quelli con codice superiore a 127 come se fossero tokens.

COPY seguito da EXP esegue sulla stampante Epson-compatibile collegata una copia dello schermo a scale di grigi. Aggiungendo a COPY EXP anche INVERSE si ottiene lo hard copy grey scale con l'ink invertito con il paper (utile per risparmiare inchiostro con disegni dallo sfondo scuro). E' inutile ricordare che la vecchia ZX Printer non rientra nei programmi della Sinclair gestione Amstrad.



Conclusioni

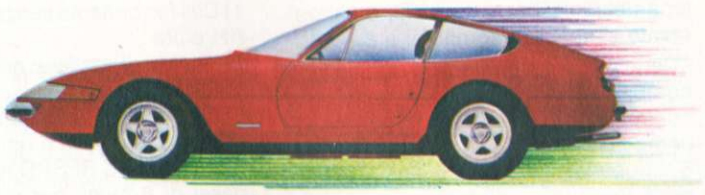
Il +3 è sicuramente una bella macchina, completa e destinata ad avere successo se aiutata da una adeguata politica da parte della Amstrad/Sinclair. Il prezzo in moneta inglese infatti non dovrebbe superare le 250 sterline (che di per sè è già abbastanza). Una nota negativa è l'incompatibilità con il +2 e la conseguente impossibilità di aggiungere al +2 il solo disco senza dover comprare l'intero computer.

Il manuale, scritto sulla base del testo originale di Steven Vickers "ZX Spectrum Basic programming", è completo e pronto a rispondere ad ogni vostro quesito (vedete di leggerlo dunque!) e comprende anche gli entry points di tutte le routines del +3 DOS.

L'ultima considerazione riguarda proprio i dischi a 3" dello Spectrum: essendo gli stessi che vengono utilizzati dagli altri computer Amstrad, si può sperare che le software houses mettano in commercio dischi con su di un lato la versione Amstrad dei loro programmi e sull'altro la versione Spectrum, il che potrebbe far diminuire i prezzi.

METTI IL TURBO AL TUO QL

a cura di Eros Forenzi



Il compilatore Turbo della software house inglese Digital Precision è probabilmente "IL" programma che molti aspettavano.

Pensate un po': riuscire ad ottenere un programma execabile, multitasking, con completa gestione degli errori, di lunghezza anche di 640K, velocissimo, senza più i caricamenti a singhiozzo dei programmi Superbasic, e, quel che conta sopratutto, senza conoscere alcunchè di linguaggio macchina 68000.

Turbo è tutto questo e molto altro.

E' molto difficile sintettizzare in un solo articolo tutto un manuale di 350 pagine; ma non spaventatevi: il manuale è molto più semplice e divertente del disassemblato di una rom qualsiasi, ed è intercalato qua e là dalla giusta dote di umorismo necessaria a renderne meno noiosa la lettura.

Questo prodotto viene venduto in una confezione composta da due microdrive oppure da un singolo floppy da 3 e mezzo, accompagnata da due manuali.

Eh sì, i manuali sono due: quello del compilatore e quello di Turbo Toolkit.

Questo toolkit è una estensione del Superbasic di 6 k di lunghezza... certo che definirlo una semplice estensione è riduttivo. Sulla cartuccia del toolkit trovate infatti molte altre cose, programmini accessori per il compilatore ma anche due simpatici regali: un editor di caratteri e un sound editor, naturalmente realizzati con Turbo.

Questo toolkit deve essere caricato in memoria prima di poter compilare un qualsiasi programma Superbasic. A questo punto caricate il programma basic da compilare e poi inserite nel mdv1 la cartuccia contenente Turbo.

Ora siete pronti per partire.

Digitate il comando CHARGE e aspettate una decina di secondi; apparirà il menù principale di Turbo da dove potrete con pochissime scelte dare il via alla compilazione del vostro programma basic.

Il procedimento è il seguente:

1) inserire il nome del file compilato.

2) inserire il nome del file sorgente (cioè il nostro programma in basic, che potremo scaricare su mdv- oppure semplicemente listare sullo schermo man mano che la compilaziona procede)

3) scegliere quale opzione del compilatore azionare (dobbiamo cioè decidere se vogliamo un codice compilato che abbia la maggior velocità possibile oppure la minor occupazione di memoria, oppure entrambe e così via).

4) a questo punto potete bervi tranquillamente una tazza di tè;

Turbo farà il resto del lavoro, convertire cioè il vostro programma Superbasic, lento, lungo nel caricamento, e poco flessibile in puro linguaggio macchina 68000.

Ora togliete le cartuccine, resettate il computer e ricaricate il vostro nuovo programma compilato.

No, non c'è bisogno di nessun toolkit, quello serve solo prima della compilazione e non più dopo, a patto naturalmente che non abbiate usato nessuno dei settanta e oltre comandi aggiuntivi in esso contenuti.

Potete anche caricare il vostro programma su un altro QL, che so, un JM o un JS o un MG o un AH; funzionerà perfettamente.

Anche la cartuccia contenente il compilatore è piena zeppa di files: tutto l'occorrente per rimediare a molti piccoli "inconvenienti" del QL.

Ad esempio chi ha un QL base di 128K di memoria può avere in memoria assieme al compilatore un programma di lunghezza non superiore a 20 k.

"E allora io che ho appena terminato un fantastico invaders in basic lungo 80 k come faccio?". Calma: c'è un programmino apposito di nome MAKE-MODULES che provvederà automaticamente a dividere in tanti pezzi il tuo invaders; poi dovrai compilarle separatamente ma alla fine avrai 4 files execabili da caricare uno dopo l'altro che ti daranno il brivido degli invaders superveloci.

Turbo compila quasi tutto; dico "quasi" perchè non digerisce la spazzatura, ossia i listati pieni di errori o con tecniche di programmazione talmente contorte che anche un umano di buona volontà rinuncerebbe a comprendere.

Non vengono accettati comandi del tipo LIST, RUN, RENUM, EDIT, che come potete ben capire non hanno molto senso all'interno di un programma compilato. Turbo è comunque in grado di correggere automaticamente errori minori di programmazione, come ad esempio un END DEFINE dimenticato o un DEF PROC di troppo.

Accidenti, le cose da dire sono talmente tante che... si corre il rischio di fare una grossa confusione.

Parliamo un po' di velocità: Turbo batte la concorrenza senza problemi (sia come velocità di compilazione sia come velocità del programma compilato), un semplice ciclo

FOR I=1 TO 10000

viene eseguito in 2 secondi contro i 19-20 del basic.

In genere dovete aspettarvi miglioramenti in velocità di 3, 6, 9 volte rispetto al basic.

Vi pare poco? Può darsi, ricordatevi però che Turbo compila tutto il Superbasic... ma non solo quello. Potete compilare anche un programma contenete altre estensioni al Superbasic, come quelle del Gigabasic o del Graphic Toolkit, o di Toolikit 2... praticamente tutto.

Inoltre Turbo si confronta molto bene anche con gli altri computer e compilatori presenti sul mercato.

E' più veloce dell'Amiga Basic, è più veloce del Quickbasic per IBM, e naturalmente strapazza i computer a 8 bit.

Certo non è il più veloce in assoluto tra i compilatori. Ad esempio è battuto nel confronto con alcuni prodotti per Atari St come il GFA basic & compiler o l'HISOFT ST compiler o come il FAST BASIC per ST.

Ma non tutto è perduto; il QL non ha infatti il 68000 a 16 bit dell'amico d'oltreoceano ..... e poi (quel che più conta) l'ST non è un computer multitasking.

E per finire, notizia delle notizie, è appena uscita in Inghilterra la versione 2.0 di Turbo, io vi ho parlato fin'ora della 1.09, che a giudicare dalla pubblicità sempre veritiera della Digital Precision dev'essere veramente fantastica.

Dati tecnici

Compilatore **TURBO** prodotto dalla **Digital Precision**
Funziona su QL AH, JM, JS, MG in versione base o espansa, con dischi o microdrive.
Prezzo : **99 sterline**
Manuale : **250 pagine**
Turbo viene venduto in una confezione contenete anche il programma **TURBO TOOLKIT** sempre prodotto dalla Digital Precision.
Manuale di Turbo Toolkit : **100 pagine**
Versioni di Turbo prodotte fin'ora: 1.09, 1.12, 1.13, 1.14, 2.0
Versioni di Turbo Toolkit : 1.19, 1.32, 1.37, 1.42, 2.0
Avvertenze : Turbo funziona solo con le versioni di T.Toolkit dalla 1.19 esclusa in avanti.

Caratteristiche software

• Gestione completa degli errori su qualsiasi tipo di rom e in maniera molto più completa ed efficiente della When Error contenuta sulle rom JS e MG.
• Possibilità di costruire cicli For Next con variabili intere, per ottenere notevoli aumenti in velocità, (ricordo che fino a ieri il QL trattava tutte le variabili in floating point).
• Select on stringa.
E' ad esempio possibile fare una cosa del genere:

```
100 INPUT A$
110 SELECT ON A$
120  ON A$='tuo nome'PRINT 'esatto'
130 END SELECT
```

• Moltissimi bug del QL vengono eliminati. Alcuni esempi:
1) Call funzionante senza problemi su QL AH e JM
2) Buffer di tastiera non più limitato a 128 bytes
3) Cifre numeriche visualizzate con nove cifre di precisione e non più sette.
4) Il comando RESPR alloca memoria a passi di 8 bytes per volta e non più a blocchi di 512 come invece fa sotto l'interprete Superbasic.
5) Molti altri bug secondari vengono corretti da Turbo. Alcuni rivestono importanza solo per il programmatore evoluto mentre altri sono di interesse più generale. Si tratta di bug dalla spiegazione molto tecnica, per cui rimandiamo ad un prossimo articolo in cui parleremo esclusivamente dei bugs del sistema QL.

• Velocità dei programmi compilati:
Turbo produce vero codice macchina 68000, e in alcuni casi si dimostra migliore di routines prodotte a mano da programmatori professionisti.
I miglioramenti in velocità non sono galattici nel caso di programmi in Superbasic classici (ossia scritti per funzionare con l'interprete e non per essere compilati).
Turbo non può far molto nei casi in cui c'è un pesante uso di routines della rom (ad esempio una serie di print sullo schermo) oppure quando si impiegano i microdrives in modo massiccio.
Turbo offre però la possibilità di creare programmi impossibili da far girare sotto l'interprete ma che sfruttano al massimo tutte le facility offerte dal compilatore oppure qualcuno dei potentissimi comandi aggiuntivi contenuti nel Turbo Toolkit.
In questo modo si ottengono miglioramenti in velocità incredibili. Una semplice cancellazione di schermo si può fare con Turbo in cinque o sei modi diversi dal classico comando CLS, con risultati da mozzare il fiato.

• Differenze tra le varie versioni di Turbo:
In generale si tratta di piccoli miglioramenti nella velocità del programma compilato oppure di soluzione di problemi di compatibilità con altri programmi commerciali.
Ad esempio le versioni dalla 1.14 in poi sono compatibili con TASKMASTER, QSWITCH, con il front end QRAM e con il computer THOR.
La versione 2.0 (appena uscita) pare essere veramente una bomba, sia come velocità che per opzioni per condizionare il codice compilato (Almeno 200).

• La Digital Precision non pone alcuna restrizione sulla commercializzazione di programmi compilati con Turbo. Chiede solo che venga scritto ben chiaro sulla confezione che quel programma è stato compilato con Turbo.

• Non possiamo non ringraziare coloro che hanno creato questo fantastico programma. Sono Simon Goodwin (già autore di Supercharge), Jerry Jackson e Chas Dillon.

Due parole su Turbo Toolkit:
Venduto assieme a Turbo (è indispensabile averlo in memoria prima di compilare alcunchè), oppure separatamante, questo toolkit si compone di circa 70 comandi aggiuntivi per il Superbasic. Questa estensione non va ad interferire con altri prodotti analoghi ma è intesa come complemento di essi.
La maggior parte dei comandi è stata concepita per essere utilizzata all'interno di programmi compilati, dove viene sfruttata al massimo.
Purtroppo non è molto facile ottenere risultati immediati se siete dei programmatori in erba. Gli esperti troveranno invece quello che hanno sempre sognato.
Con Turbo e Turbo Toolkit il QL non diventa solo un computer multitasking ma acquista anche la possibilità di comunicare tra tasks che funzionano in uno stesso momento. Non più tanti piccoli compartimenti stagni che vivono ognuno di vita propria, ma parti modulari e interattive di un unico grande programma.
Le potenzialità nascoste sfuggono agli stessi ideatori del toolkit, "Solo il tempo potrà dire cosa saremo stati capaci di ottenere con Turbo e Turbo Toolkit ", dicono.
Oltre all'estensione basic, il toolkit comprende anche un file basic di 41k di lunghezza composto di procedure che svolgono i compiti più disparati. Sono in tutto circa 170.
Ci sono poi due utility molto.....utili; un sound editor un editor di caratteri.

Un'ultimissima cosa : Turbo e Turbo Toolkit non sono protetti contro la copia. Niente più lenslock nè cartucce "master" nel mdv2- dunque.
Arrivederci e..... buon lavoro.

OTHELLO

di Matteo Bertolini

Versione BASIC del famosissimo gioco da tavolo: è possibile giocare sia contro un avversario "umano" (in questo caso lo Spectrum si limita a controllare la validità delle mosse e a tenere il punteggio) che contro l'intelligenza indotta nella CPU dal presente programma.

Le modalità operative sono semplicissime: per posizionare la propria pedina basta dare le coordinate al computer (prima la 'x', poi la 'y'), e inpiù sono disponibili le seguenti opzioni (da inserire al posto della mossa):

- s : serve a visualizzzare il numero di pedine in possesso di ciascun giocatore.
- p : serve per passare la mossa all'avversario in caso non si sia in condizioni di muovere.
- c : stampa la copia dello schermo.
- m : memorizza la partita con la situazione presente sullo schermo.
- l : carica una partita precedentemente memorizzata con il comando "m".

Il computer è inoltre in grado di rilevare l'eventuale situazione contemporanea di stallo di entrambi i giocatori, oltre agli errori dovuti a mosse non consentite.

Come si gioca a Othello? Per i venticinque lettori che non lo sapessero, lo scopo del gioco è quello di imprigionare i dischi avversari, il che da luogo al loro capovolgimento, che comporta il fatto che essi cambino colore e assumano quello in proprio possesso (sono bicolori). "Imprigionare" significa fare in modo che tra il disco appena posato sulla scacchiera e un altro proprio disco ci siano dei dischi avversari sulla stessa fila o colonna. All'inizio bisogna porre due dischi ciascuno sulle 4 caselle centrali della scacchiera, facendo in modo che caselle dello stesso colore si trovino in diagonale e non sulla stessa fila o colonna. Vince chi alla fine avrà più dischi del suo colore sulla scacchiera. Strategie ? In 5 minuti si è già campioni. Divertimento ? Tanto.

(List a pag. XIII)

DELETE FILES

di Ezio Boscani

del-files2 è una semplice utility che permette di fare ordine, cancellando files indesiderati, nelle vostre cartucce per il mdrive Spectrum.

Il programma si compone di una parte Basic e di due routines in L/M necessarie la prima, SHP, per scrivere il contenuto di una stringa come caratteri stampabili o, se non stampabili o con codice maggiore di 127, in notazione esadecimale in INV VIDEO; la seconda, VAL$ #c, per eseguire un catalogo della cartuccia nel drive 1 (default) completo, cioè che permette di vedere anche i files che hanno nome iniziante con CHR$(0) e il catalogo andrà al canale specificato nel numero dopo il #: Se il canale è chiuso, verrà stampato l'errore O Invalid stream.

La routine e rivolta verso coloro che sono in possesso del Mdrive Spectrum ed è strutturata in modo user-friendly, cioè semplice da capire.

Comunque le istruzioni sono:

- Tasti 6/7 : spostano il cursore su/giu lungo la lista di files, che può essere anche più lunga di una schermata;
- Tasti S/D : scelta o non scelta del file da cancellare: il file scelto si troverà un asterisco vicono al nome; se si preme D col cursore presso tale file, l'asterisco si cancellerà ed anche il file dalla lista;
- Tasto E : Inizia le operazioni di cancellazione;
- Tasto 0 : Abortisce l'operazione e fa ripartire il programma dall'inizio.

La routine SHP può essere utilizzata separatamente dal programma ed è completamente rilocabile; una volta caricata, per farla agire, si usa la sintassi RAND a$=STR$ USR indirizzo.

La routine VAL$ #c è fissa all'indirizzo 64000 per 200 bytes, e si attiva, una volta caricata, con RAND USR 64000.

(List a pag. XVII)

CALCOLO BALISTICO

programma di Gianluigi Perrella

Ottimo e completo programma per l'analisi del moto dei punti materiali, trascurando gli attriti.

Il programma, sulla base di una serie di dati inseriti dall'utente (quali ad esempio Velocità iniziale, angolo con l'orizzontale,..), calcola e visualizza l'equazione della traiettoria del punto (che, con le opportune approssimazioni, può essere benissimo un sasso, una palla..), la gittata, e permette anche di cambiare i singoli valori iniziali in modo da mostrare i cambiamenti nella traiettoria che ciò comporta. Ad alcune richieste di dati è anche consentito non rispondere, lasciando al computer la decisione di stabilire se il dato è necessario o meno per il calcolo desiderato.

I calcoli effettuati dal programma sono:

- equazione della traiettoria
- x(t) e y(t)
- velocità iniziale
- angolo di lancio
- t, dati x(t) e y(t)
- velocità verticale e orizzontale
- quota massima raggiunta
- tempo per raggiungere una data quota
- tempo totale di volo
- gittata massima

(List a pag. IXX)

IL PROBLEMA DELLE PROVE RIPETUTE

di Gianluca Padovani e Mario Bianchi

Quando si prende in esame un qualsiasi elemento che presenta delle caratteristiche si guarda sempre quante e quali di si possono presentare e con che frequenza si presentano.

Gli eventi che vengono presi in esame sono teoricamente degli eventi aleatori e la loro probabilità è maggiore a seconda del numero delle cause che possono produrli o impedirli. Possiamo inoltre dividere due diversi tipi di probabilità che possono definire un dato evento: la prima è quella che può essere calcolata matematicamente e che ci da un idea ben precisa del tipo di fenomeno da studiare; la seconda e quella che non è definibile da un solo calcolo matematico a causa dei molteplici fattori che caratterizzano l'evento e che si puo desumere solo con l'esperienza.

Si guarda inoltre se gli eventi studiati hanno una certa compatibilità e quindi se dei determinati caratteri che li definiscono hanno delle connesioni.

Guardiamo ora ad uno specifico caso in cui si applica il calcolo delle probabilità nel quale si cerca di stabilire se un dato evento dipende più o meno dal verificarsi contemporaneamente o in succesione, di due o più eventi compatibili o indipendenti. E' appunto da questo studio che viene detto "calcolo della probabilità composta", che si stacca un caso particolare che andiamo ora ad analizzare riguardante appunto le prove ripetute.

Prendiamo quindi in analisi un dato elemento che presenta una probabitità costante (P) calcolata teoricamente. Sottoponiamo il nostro evento ad un certo numero di prove (N) e cerchiamo di stabilire qual è la probabilità che su queste n prove l'evento si verifichi un certo numero di volte (K).

Ad esempio consideriamo un dado di cui conosciamo già la probabilità che da un lancio esca una determinata faccia (1/6). Lanciando però il dado più di una volta diventa complicato stabilire quale sia la probabilità che si presenti una determinata faccia una o più volte ed è appunto per questo motivo che si utilizza il metodo delle prove ripetute.

Riprendendo in esame il dado, se vogliamo conoscere qual è la probabilità che in tre lanci la faccia prefissata non potrà uscire mai dobbiamo innanzi tutto tenere conto delle probabilità contraria che si verifichi una data faccia.

Questa si calcola sapendo che le facce di una dado sono 6 e che si possono presentare tutte tranne una prescelta (Q= 5/6 dove Q è la probabilità contraria Q=1-P).

La probabilità che su tre lanci la data faccia non si presenti mai sarà:

5/6 * 5/6 * 5/6 = Q 3

La probabilità che su tre lanci compaia solo al primo lancio sarà:

1/6 * 5/6 * 5/6 = P * Q 2

Se infine si vuole la probabilità che su tre la nci compaia sempre tre volte si avrà:

1/6 * 1/6 * 1/6 =P 3

Estendendo il ragionamento al caso generale di n prove si ricaverà la regola generale che permetterà di conoscre la probabilità che un qualsiasi evento si possa presentare K volte so N prove.

La probabilità sara quindi data dalla formula:

Pk= (n k) * P k * Q n-k

dove *n* è il numero delle prove, *k* il numero delle volte *P* e *Q* le probabilità che l'evento si verifichi.

Il programma

Sicuramente gli esempi fino ad ora fatti non richiedono certo l'uso di un programma come quello che presentiamo, ed in effetti nella maggir parte dei casi questi calcoli possono anche essere svolti semplicemente con carta e penna.

Se però si incominciano a prendere degli eventi le cui probabilità non sono più semplici frazioni o il numero delle ripetizioni diventa abbastanza grosso, i calcoli non appaiono più così semplici e l'uso di un supporto che semplifichi i calcoli diventa indispensabile.

Questo è appunto il compito del nostro programma che permette tutte le varie combinazioni di calcolo necessarie ai fini di un corretto metodo di lavoro.

La struttura del programma si presenta decisamente lineare e di facile apprendimento grazie ad una corretta serie di commenti che lo suddividono in blocchi chiave.

Il suo funzionamento è molto semplice una volta impadroniti delle varie tecnice di pianificazione dei calcoli già descritte in precedenza, ma soprattutto per la struttura stessa del programma

Le opzioni presenti sono sei di cui quattro per le modalità di calcolo, una per il cambio dei dati e una per la visualizzazione del grafico di probabilità.

Modalità di calcolo

Queste quattro opzioni permettono di selezionare tutte le combinazioni possibili di calcolo che si possono effettuare per mezzo delle operazioni già spiegate.

- La prima (K SUCCESSI) permette di calcolare la probabilità che l'evento si verifichi K volte di seguito.
- La seconda (ALMENO K SUCCESSI) calcola la probabilità che l'evento si verifichi almeno k volte su n prove partendo dall'elemento K+1 in poi.
- La terza (FINO A K SUCCESSI) calcola la probabilità totoale che l'evento si verifichi al massimo k volte su n.
- L'ultima (DA K A K1 SUCCESSI) calcola la probabilità che l'evento si verifichi K1-k volte su n partendo da K fino a K1.

Cambio dati e grafico

Queste due opzioni sono necessarie per un utilizzo più flessibile dell'intero programma che richiede un continuo aggiornamento dei dati senza però penalizzare la velocità di esecuzione del lavoro globale.

I dati riguardanti la probabilità sono sempre visualizzati nel menù principale mentre la richiesta del parametro K o K1 viene inoltrata dopo il selezionamento dell'opzione.

Per quanto riguarda l'ultima opzione del menù principale, cioè quella di visualizzazione del grafico non vi è una specifica connessione fra le opzioni precedenti e quest'ultima.

Variando la probabilità dell'evento, varierà la pendenza del grafico che rispecchia l'andamento della distripuzione binomiale con la classica evoluzione a campana.

Dopo aver quindi battuto il listato con attenzione non rimane che dare il classico RUN e lanciarsi nel mondo statistico con il problema delle prove ripetute.

(List a pag. XXIII)

COMMODORE

PASSEGGIANDO TRA I BANCHI

```
3 : REM LISTATO 1
4 :
10 SCNCLR: PRINT SPC(8);
15 PRINT"VALORE IN 65280 ($FF00)":
   PRINT
20 PRINT"BANK"," DEC","HEX":PRINT
30 FOR X=0 TO 15
40 SYS DEC("F7EC"),,X
50 PRINT X,PEEK(6),"$";
55 PRINT RIGHT$(HEX$(PEEK(6)),2)
60 NEXT
```

```
3 : REM LISTATO 2
4 :
10 SCNCLR:A$="PREMI UN TASTO"
20 PRINT CHR$(18)" DOPO IL RUN ";
25 PRINT CHR$(146)
30 GOSUB50:FORX=0TO15
35 PRINTCHR$(18)"BANCO";X;
40 PRINTCHR$(146):BANKX
45 GOSUB50:NEXT:END
50 PRINT:PRINT" REGISTRO $FF00 = $";
55 PRINT HEX$(PEEK(65280));
60 PRINT " ("PEEK(65280)")"
65 PRINT:PRINT:PRINTA$:GETKEYB$
70 SCNCLR:RETURN
```

```
3 : REM LISTATO 3
6 :
10 SCNCLR
20 FORX=16831TO16976
30 A=PEEK(X):IFA=64THENA=0
40 PRINTCHR$(A);:NEXT
```

```
3 : REM LISTATO 4
6 :
10 FORX=16831TO16976
20 BANK15:A=PEEK(X):IFA=64THENA=0
30 BANK1:POKEX-8639,A:NEXT
40 PRINTHEX$(PEEK(65280)):STOP
50 SCNCLR:FORX=8192 TO 8336
60 PRINTCHR$(PEEK(X));:NEXT
70 BANK 15:END
```

```
3 : REM LISTATO 5
6 :
10 FOR X=4864TO4896:READA:U=U+A
15 POKE X,A: NEXT
20 IF U=3492 THEN SYS4864
30 DATA 120,169,013,162,019,141
40 DATA 020,003,142,021,003,088
50 DATA 096,162,038,189,190,065
60 DATA 201,064,048,002,233,064
70 DATA 157,000,004,202,208,241
80 DATA 076,101,250
90 END
```

```
3 : REM LISTATO 6
6 :
10 BANK0
20 FOR X=1 TO 16:READ A:U=U+A
25 POKE 652+X,A:NEXT
30 FOR X=1 TO 48:READ A:U=U+A
35 POKE 49151+X,A: NEXT
40 IF U=7225 THEN SYS 49152
50 REM---------- ROUTINE IN $28D
60 DATA 169,063,141,000,255,076
70 DATA 013,192,169,000,141,000
80 DATA 255,076,101,250
90 REM--------- ROUTINE IN $C000
100 DATA 120,169,141,162,002,141
110 DATA 020,003,142,021,003,088
120 DATA 096,169,190,162,065,133
130 DATA 250,134,251,160,038,162
140 DATA 000,169,250,141,170,002
150 DATA 032,162,002,201,064,048
160 DATA 002,233,064,153,000,004
170 DATA 136,208,234,076,149,002
180 END
```

```
3 : REM LISTATO 7
6 :
100 FORX=1TO16:READA:U=U+A
```

```
105 POKE652+X,A:NEXT
110 BANK0:FORX=1TO13:READA:U=U+A
115 POKE49151+X,A:NEXT
120 BANK1:FORX=1TO50:READA:U=U+A
125 POKE49164+X,A:NEXT
130 IFU<>9346THENPRINT"ERRORE":END
135 SCNCLR:BANK0:SYS49152
150 REM ------------------------
170 DATA 169,127,141,000,255,076
180 DATA 013,192,169,000,141,000
190 DATA 255,076,101,250
210 REM-------------------------
230 DATA 120,169,141,162,002,141
240 DATA 020,003,142,021,003,088
250 DATA 096
270 REM-------------------------
290 DATA 169,190,162,065,133,250
300 DATA 134,251,169,000,162,004
310 DATA 133,252,134,253,160,038
320 DATA 162,000,169,250,141,170
330 DATA 002,032,162,002,201,064
340 DATA 048,002,233,064,162,252
350 DATA 142,185,002,162,000,032
360 DATA 175,002,136,208,227,076
370 DATA 149,002
380 END
```

DISASSEMBLATO 1

```
01300 SEI           ;Disabilita le interruzioni.
01301 LDA #$0D      ;Inserisce l'indirizzo
01303 LDX #$13      ;$130D nel vettore di
01305 STA $0314     ;interrupt (locazioni
01308 STX $0315     ;$314 e $315).
0130B CLI           ;Riabilita le interruzioni.
0130C RTS           ;Return.
                    ;-------------------------
0130D LDX #$26      ;Numero dei caratteri in X.
0130F LDA $41BE,X   ;Ne preleva uno da $41BE+X.
01312 CMP #$40      ;Confr. il suo ASCII con 64.
01314 BMI $1318     ;Se minore salta a $1318.
01316 SBC #$40      ;Sottrae 64 a valore ASCII e
01318 STA $0400,X   ;lo invia sullo schermo.
0131B DEX           ;X=X-1
0131C BNE $130F     ;Se X <> 0 salta a $130F.
0131E JMP $FA65     ;Va all'interrupt del S.O.
```

DISASSEMBLATO 2 (START a $C000 di bank 0)

```
0028D LDA #$3F      ; Seleziona il banco 0
0028F STA $FF00     ; e salta a $C00D.
00292 JMP $C00D     ; ------------------------------
00295 LDA #$00      ; Seleziona il banco 15
00297 STA $FF00     ; e salta alla routine
0029A JMP $FA65     ; di interrupt.

==================================================

0C000 SEI           ;
0C001 LDA #$8D      ;  Dirotta l'interrupt
0C003 LDX #$02      ;
0C005 STA $0314     ;  all'indirizzo $28D
0C008 STX $0315     ;
0C00B CLI           ;    (area comune)
0C00C RTS           ; ---------------------------------
```

```
0C00D LDA #$BE      ; Prepara le locazioni di pagina
0C00F LDX #$41      ; zero $FA e $FB  con
0C011 STA $FA       ; l'indirizzo iniziale da cui
0C013 STX $FB       ; prelevare il contenuto.
0C015 LDY #$26      ; Numero dei caratteri in Y,
0C017 LDX #$00      ; Valore per numero di banco in X.
0C019 LDA #$FA      ; Mette in $2AA il valore del
0C01B STA $02AA     ; primo puntatore di pagina zero.
0C01E JSR $02A2     ; Salta a INDFET.
0C021 CMP #$40      ; ---------------------------------
0C023 BMI $C027     ;  Invia nelle locazioni di schermo
0C025 SBC #$40      ;  il valore ASCII meno 64 del
0C027 STA $0400,Y   ;  carattere prelevato (presente
0C02A DEY           ;  in accumulatore) da INDFET.
0C02B BNE $C017     ; ---------------------------------
0C02D JMP $0295     ;  Salta in area comune.
```

DISASSEMBLATO 3 (START a $C000 di bank 0)

```
0028D LDA #$7F      ;  Seleziona il banco 1
-----------------------------------------

0028F - 0029C       ;     IDENTICO  AL
0C000 - 0C00C       ;   DISASSEMBLATO 2

=========================================


1C00D LDA #$BE      ;  Alloca indirizzo $41BE
1C00F LDX #$41      ;  in due locazioni di
1C011 STA $FA       ;  pagina 0 per INDFET.
1C013 STX $FB       ; ----------------------

1C015 LDA #$00      ;  Alloca indirizzo $0400
1C017 LDX #$04      ;  in due locazioni di
1C019 STA $FC       ;  pagina 0 per INDSTA.
1C01B STX $FD       ; ----------------------
1C01D LDY #$26      ;  Legge i caratteri
1C01F LDX #$00      ;
1C021 LDA #$FA      ;  tramite INDFET
1C023 STA $02AA     ;
1C026 JSR $02A2     ; ----------------------
1C029 CMP #$40      ;  Sottrae 64 al valore
1C02B BMI $C02F     ;  ASCII del carattere.
1C02D SBC #$40      ; ----------------------
1C02F LDX #$FC      ;  Deposita il valore (in
1C031 STX $02B9     ;  accumulatore) nell'area
1C034 LDX #$00      ;  schermo di bank 15
1C036 JSR $02AF     ;  tramite INDSTA.
1C039 DEY           ;
1C03A BNE $C01F     ; ----------------------
1C03C JMP $0295     ;  Salta in area comune
```

QL SUPER WINDOW

```
100 :
110 REMark * * * * * * * * * * * * * *
120 REMark *                         *
130 REMark *  GRAF_CODE GENERATOR    *
140 REMark *    (C) MCMLXXXVII By    *
150 REMark *      SUD SOFTWARE       *
160 REMark *    MOTTA S. LUCIA CZ    *
170 REMark *                         *
180 REMark *  NOTARIANNI CARLO       *
190 REMark *                         *
200 REMark * * * * * * * * * * * * * *
210 :
220 RESTORE 400:A=RESPR(900)
230 BASE=A:COUNT=0
240   REPeat L
250     IF EOF:EXIT L
260     CHECK=0
270       FOR I=0 TO 5
280         READ X
290         POKE A,X
300         A=A+1
310         CHECK=CHECK+X
320       END FOR I
330     READ CHECKSUM
340     IF CHECK<>CHECKSUM :BEEP 3200
0,0:PRINT 'ERRORE ALLA LINEA ';PEEK_W
(PEEK_L(163856)+252):STOP
350     COUNT=COUNT+6
360   END REPeat L
370 :
380 SBYTES mdv1_GRAF_code,BASE,COUNT
390 :
400 DATA 67,250,0,12,52,120,501
410 DATA 1,16,78,146,112,0,353
420 DATA 78,117,0,2,0,54,251
430 DATA 9,83,67,82,95,83,419
440 DATA 84,79,82,69,1,4,319
450 DATA 11,83,67,82,95,82,420
460 DATA 69,70,82,69,83,72,445
470 DATA 0,0,0,1,1,206,208
480 DATA 8,83,67,82,95,83,418
490 DATA 73,90,69,0,0,0,232
500 DATA 0,0,52,120,1,24,197
510 DATA 78,146,78,117,8,54,481
520 DATA 0,7,184,1,103,10,305
530 DATA 97,0,2,44,74,128,345
540 DATA 102,238,96,40,97,228,801
550 DATA 74,128,102,230,12,67,613
560 DATA 0,1,103,0,2,168,274
570 DATA 12,67,0,5,102,92,278
580 DATA 34,54,152,0,36,54,330
590 DATA 152,4,38,54,152,8,408
600 DATA 40,54,152,12,42,118,418
610 DATA 152,16,187,252,0,2,609
620 DATA 0,0,109,64,97,66,336
630 DATA 74,128,102,182,58,193,737
640 DATA 58,194,58,195,58,196,759
650 DATA 2,129,0,0,3,255,389
660 DATA 12,65,0,8,108,2,195
670 DATA 114,8,228,137,8,129,624
680 DATA 0,0,83,65,83,66,297
690 DATA 44,1,26,217,81,201,570
700 DATA 255,252,215,252,0,0,974
710 DATA 0,128,34,75,50,6,293
720 DATA 81,202,255,238,112,0,888
730 DATA 78,117,112,241,78,117,743
740 DATA 12,65,0,2,109,74,262
750 DATA 12,65,2,0,110,68,257
760 DATA 12,66,0,2,109,62,251
770 DATA 12,66,1,0,110,56,245
780 DATA 74,67,109,52,60,60,422
790 DATA 2,0,156,65,188,67,478
800 DATA 109,42,74,68,109,38,440
810 DATA 60,60,1,0,156,66,343
820 DATA 188,68,109,28,34,124,551
830 DATA 0,2,0,0,44,3,49
840 DATA 228,142,8,134,0,0,512
850 DATA 211,198,44,4,239,142,838
860 DATA 211,198,38,73,112,0,632
870 DATA 78,117,112,252,78,117,754
880 DATA 97,0,255,28,74,128,582
890 DATA 102,158,12,67,0,1,340
900 DATA 109,154,12,67,0,4,346
910 DATA 110,148,42,118,152,0,570
920 DATA 187,252,0,2,0,0,441
930 DATA 109,136,12,67,0,1,325
940 DATA 102,12,122,0,50,29,315
950 DATA 52,29,54,29,56,29,249
960 DATA 96,54,12,67,0,2,231
970 DATA 102,6,42,54,152,4,360
980 DATA 96,234,12,67,0,3,412
990 DATA 102,18,122,0,38,54,334
1000 DATA 152,4,40,54,152,8,410
1010 DATA 50,29,52,29,46,29,235
1020 DATA 96,18,42,54,152,4,366
1030 DATA 38,54,152,8,40,54,346
1040 DATA 152,12,50,29,52,29,324
1050 DATA 46,29,97,0,255,66,493
1060 DATA 74,128,102,0,255,54,613
1070 DATA 2,129,0,0,3,255,389
1080 DATA 12,65,0,8,108,2,195
1090 DATA 114,8,228,137,8,129,624
1100 DATA 0,0,83,65,83,66,297
1110 DATA 44,1,74,133,103,4,359
1120 DATA 106,24,107,46,18,221,522
1130 DATA 81,201,255,252,215,252,1256
1140 DATA 0,0,0,128,34,75,237
1150 DATA 50,6,81,202,255,238,832
1160 DATA 96,46,24,29,137,25,357
1170 DATA 81,201,255,250,215,252,1254
1180 DATA 0,0,0,128,34,75,237
1190 DATA 50,6,81,202,255,236,830
1200 DATA 96,22,24,29,185,25,381
1210 DATA 81,201,255,250,215,252,1254
1220 DATA 0,0,0,128,34,75,237
1230 DATA 50,6,81,202,255,236,830
1240 DATA 112,0,78,117,187,203,697
```

```
1250 DATA 103,0,1,68,8,54,234
1260 DATA 0,7,184,1,102,0,294
1270 DATA 1,26,97,0,254,48,426
1280 DATA 74,128,102,102,12,67,485
1290 DATA 0,2,103,10,12,67,194
1300 DATA 0,4,103,18,96,0,221
1310 DATA 254,166,34,54,152,0,660
1320 DATA 36,54,152,4,118,0,364
1330 DATA 120,0,96,16,34,54,320
1340 DATA 152,0,36,54,152,4,398
1350 DATA 38,54,152,8,40,54,346
1360 DATA 152,12,97,0,254,136,651
1370 DATA 74,128,102,48,12,65,429
1380 DATA 0,8,108,2,114,9,241
1390 DATA 228,137,194,194,80,129,962
1400 DATA 34,110,0,88,93,137,462
1410 DATA 52,60,8,32,83,66,301
1420 DATA 227,129,104,250,226,145,108
1
1430 DATA 61,130,152,0,45,129,517
1440 DATA 152,2,45,73,0,88,360
1450 DATA 120,2,112,0,78,117,429
1460 DATA 56,120,1,24,78,148,427
1470 DATA 74,128,102,244,12,67,627
1480 DATA 0,2,102,0,254,60,418
1490 DATA 32,54,152,0,42,118,398
1500 DATA 152,4,187,252,0,2,597
1510 DATA 0,0,109,0,254,42,405
1520 DATA 12,128,0,0,0,255,395
1530 DATA 110,0,254,32,97,64,557
1540 DATA 74,128,102,0,254,22,580
1550 DATA 34,8,194,252,0,4,492
1560 DATA 34,65,211,249,0,2,561
1570 DATA 128,120,36,81,32,74,471
1580 DATA 213,252,0,0,0,42,507
1590 DATA 34,18,74,129,109,0,364
1600 DATA 253,248,209,252,0,0,962
1610 DATA 0,24,54,24,56,24,182
1620 DATA 50,24,52,16,97,0,239
1630 DATA 253,234,74,128,102,0,791
1640 DATA 253,224,78,117,32,110,814
1650 DATA 0,48,192,252,0,40,532
1660 DATA 209,192,177,238,0,52,868
1670 DATA 108,12,32,54,136,0,342
1680 DATA 109,6,32,64,112,0,323
1690 DATA 78,117,112,250,78,117,752
1700 DATA 42,118,152,0,187,252,751
1710 DATA 0,2,0,0,109,0,111
1720 DATA 253,176,112,1,97,0,639
1730 DATA 255,120,96,0,253,94,818
1740 DATA 52,120,1,24,78,146,421
1750 DATA 102,222,12,67,0,1,404
1760 DATA 102,0,253,150,32,54,591
1770 DATA 152,0,97,0,255,102,606
1780 DATA 74,128,102,202,96,0,602
1790 DATA 255,0,112,1,96,240,704
```

```
;       BP_INIT   EQU  $110
;       CA_GTLIN  EQU  $118

        LEA       PDEF(PC),A1       ; INIZIALIZZA I NUOVI COMANDI
        MOVEA.W   BP_INIT,A2        ; CARICA IL VETTORE
        JSR       (A2)              ; ED ESEGUI ...
        MOVEQ     #0,D0             ; NESSUN ERRORE
        RTS                         ; TORNA AL BASIC
PDEF
        DC.W      2                 ;
        DC.W      STORE-*           ;
        DC.B      9,'SCR_STORE'     ; TAVOLA DI DEFINIZIONE
        DC.W      REFRESH-*         ; DEI NUOVI COMANDI DA
        DC.B      11,'SCR_REFRESH'  ; LINKARE 2 PROCEDURE
        DC.W      0                 ; ED UNA FUNZIONE
        DC.W      1                 ; E RELATIVI PUNTATORI
        DC.W      SIZE-*            ; ALLE ROUTINES STESSE
        DC.B      8,'SCR_SIZE',0    ;
        DC.W      0,0               ;
PREL
        MOVEA.W   CA_GTLIN,A2       ; PRELEVA LONG_WORD
        JSR       (A2)              ; ESEGUI ....
OUT
        RTS                         ; RITORNA AL PUNTO CHIAMANTE
STORE
        BTST      #7,1(A6,A3.L)     ; il primo parametro e' '#' ?
        BEQ.S     BASIC_PAR         ; NO:PARAMETRI PASSATI
        BSR       FIND_CAN          ; GO SUB CERCA CANALE VIDEO
        TST.L     D0                ; ERRORI  ?
        BNE.S     OUT               ; SI: TORNALI AL BASIC
```

```
        BRA.S   POST_INPUT          ; SALTA GLI INPUT DA BASIC
BASIC_PAR
        BSR.S   PREL                ; GO SUB PRELeva parametri
        TST.L   D0                  ; ERRORI ?
        BNE.S   OUT                 ; SI: TORNALI AL BASIC
        CMPI.W  #1,D3               ; UN SOLO PARAMETRO ?
        BEQ     DEFAULT_CAN         ; GOTO DEFAULT_CAN
        CMPI.W  #5,D3               ; SONO STATI PASSATI 5 PARAMETRI
        BNE.S   BADPAR              ; NO: BADPAR
        MOVE.L   0(A6,A1.L),D1      ; LARG WINDOW
        MOVE.L   4(A6,A1.L),D2      ; ALT  WINDOW
        MOVE.L   8(A6,A1.L),D3      ; POSX WINDOW
        MOVE.L  12(A6,A1.L),D4      ; POSY WINDOW
        MOVE.L  16(A6,A1.L),A5      ; INDIRIZZO DEL BUFFER
POST_INPUT
        CMP.L   #131072,A5          ; BUFFER MINORE DI 131072 ?
        BLT.S   BADPAR              ; SI: TORNA CON ERR_BP
        BSR.S   CONTROLLA_PAR       ; GO SUB CONTROLLA_PARAMETRI
        TST.L   D0                  ; SEGNALA ERRORI ?
        BNE.S   OUT                 ; SI: TORNALI AL BASIC
        MOVE.W  D1,(A5)+            ; LARG WINDOW NEL BUFFER
        MOVE.W  D2,(A5)+            ; ALT  WINDOW NEL BUFFER
        MOVE.W  D3,(A5)+            ; POSX WINDOW NEL BUFFER
        MOVE.W  D4,(A5)+            ; POSY WINDOW NEL BUFFER
        AND.L   #1023,D1            ; PULISCI BIT SPURII DI D1
        CMPI.W  #8,D1               ; LARGhezza minima  8 ?
        BGE.S   DIVIDI              ; SI: SALTA A DIVIDI
        MOVEQ   #8,D1               ; LARGhezza minima  8
DIVIDI
        LSR.L   #2,D1               ; E DIVIDILA PER 4
         BCLR    #0,D1              ; D1 PARI  PER  FORZA
         SUBQ.W  #1,D1              ; -1 PER DBRA=NUM. BYTES RIGA
         SUBQ.W  #1,D2              ; -1 PER DBRA=NUM. DELLE COLONNE
         MOVE.L  D1,D6              ; SALVA NUMERO BYTES IN D6
 LOOP_COPIA
         MOVE.B  (A1)+,(A5)+        ; COPIA QUESTA RIGA
         DBRA    D1,LOOP_COPIA      ; D1+1 VOLTE
         ADD.L   #128,A3            ; SOMMA 128 PER NUOVA RIGA
         MOVE.L  A3,A1              ; E COPIA L'INDIRIZZO
         MOVE.W  D6,D1              ; RICOPIA LA LUNG. RIGA
         DBRA    D2,LOOP_COPIA      ; D2+1 VOLTE =NUM. COLONNE
         MOVEQ   #0,D0              ; NESSUN ERRORE
 FUORI
         RTS                        ; RITORNA
 BADPAR
         MOVEQ   #-15,D0            ; METTI ERR_BP IN D0
         RTS                        ; E RITORNA

 CONTROLLA_PAR

         CMPI.W  #2,D1              ; SUBROUTINE DI CONTROLLO DEI
         BLT.S   OUT_OF_RANGE       ; PARAMETRI PASSATI DAL BASIC
         CMPI.W  #512,D1            ; SE TUTTO OK TORNA IN A1 E A3
         BGT.S   OUT_OF_RANGE       ; L'INDIRIZZO DI INIZIO DELLA
         CMPI.W  #2,D2              ; FINESTRA NEL DISLAY FILE
```

```
        BLT.S   OUT_OF_RANGE        ; IN D1 LUNGHEZZA RIGA
        CMPI.W  #256,D2             ; IN D2 NUMERO COLONNE
        BGT.S   OUT_OF_RANGE        ; ALTRIMENTI OUT OF RANGE
        TST.W   D3
        BLT.S   OUT_OF_RANGE
        MOVE.W  #512,D6
        SUB.W   D1,D6
        CMP.W   D3,D6
        BLT.S   OUT_OF_RANGE
        TST.W   D4
        BLT.S   OUT_OF_RANGE
        MOVE.W  #256,D6
        SUB.W   D2,D6
        CMP.W   D4,D6
        BLT.S   OUT_OF_RANGE
        MOVE.L  #131072,A1
        MOVE.L  D3,D6
        LSR.L   #2,D6
        BCLR    #0,D6               ; PARI PER FORZA
        ADD.L   D6,A1
        MOVE.L  D4,D6
        LSL.L   #7,D6
        ADD.L   D6,A1               ; A1=131072+POSX/4+128*POSY
        MOVE.L  A1,A3               ; SALVA INDIRIZZO IN A3
        MOVEQ   #0,D0
        RTS
OUT_OF_RANGE
        MOVEQ   #-4,D0
        RTS
REFRESH
        BSR     PREL                ; GO SUB PRELeva parametri
        TST.L   D0                  ; ERRORI ?
        BNE.S   FUORI               ; SI: TORNALI AL BASIC
        CMPI.W  #1,D3               ; NESSUN PARAMETRO ?
        BLT.S   BADPAR              ; SI: TORNA CON ERR_BP
        CMPI.W  #4,D3               ; PIU' DI QUATTRO PARAMETRI ?
        BGT     BADPAR              ; SI :TORNA CON ERR_BP
        MOVE.L  0(A6,A1.L),A5       ; IL PRIMO PARAM. E' SEMPRE IL
        CMP.L   #131072,A5          ; BUFFER CONTROLLANE LA VALIDITA'
        BLT     BADPAR              ; SE MINORE DI 131072 ERR_BP
        CMPI.W  #1,D3               ; UN SOLO PARAMETRO ?
        BNE.S   NO_DEFAULT          ; NO VAI A NO_DEAULT
        MOVEQ   #0,D5               ; OVER 0  PER  DEFAULT
PREL_ALL
        MOVE.W  (A5)+,D1            ; PRELEVA TUTTO TRANNE OVER
        MOVE.W  (A5)+,D2
        MOVE.W  (A5)+,D3
        MOVE.W  (A5)+,D4
        BRA.S   GO_REFRESH          ; SALTA A GO_REFRESH
NO_DEFAULT
        CMPI.W  #2,D3               ; 2 PARAMETRI ?
        BNE.S   TRE_O_QUATTRO       ; NO CONTROLLA SE SONO 3 O 4
        MOVE.L  4(A6,A1.L),D5       ; SI ALLORA PRELEVA OVER
        BRA.S   PREL_ALL            ; PARAMETRI WINDOW DI DEFAULT
TRE_O_QUATTRO
```

```
        CMPI.W    #3,D3             ; SONO 3 O 4 PARAMETRI ?
        BNE.S     QUATTRO           ; SONO QUATTRO
        MOVEQ     #0,D5             ; SE TRE OVER=0
        MOVE.L    4(A6,A1.L),D3     ; PRELEVA POSX
        MOVE.L    8(A6,A1.L),D4     ; PRELEVA POSY
        MOVE.W    (A5)+,D1          ; LARG=DEFAULT
        MOVE.W    (A5)+,D2          ; ALT=DEFAULT
        MOVE.L    (A5)+,D7          ; AGGIUSTA BUFFER
        BRA.S     GO_REFRESH        ; E GO TO INIZIO
QUATTRO
        MOVE.L     4(A6,A1.L),D5    ; 4 PARAM. PRELEVA OVER
        MOVE.L     8(A6,A1.L),D3    ; PRELEVA POSX
        MOVE.L    12(A6,A1.L),D4    ; PRELEVA POSY
        MOVE.W    (A5)+,D1          ; LARG=DEFAULT
        MOVE.W    (A5)+,D2          ; ALT=DEFAULT
        MOVE.L    (A5)+,D7          ; AGGIUSTA BUFFER
GO_REFRESH
        BSR       CONTROLLA_PAR     ; GO SUB CONTROLLA
        TST.L     D0                ; ERRORI ?
        BNE       FUORI             ; SI: TORNALI AL BASIC
        AND.L     #1023;D1          ; PULISCI BIT SPURII DI D1
        CMPI.W    #8,D1             ; LARGhezza minima 8 ?
        BGE.S     DIVIDI_REF        ; SI: SALTA A DIVIDI REFresh
        MOVEQ     #8,D1             ; LARghezza minima 8
DIVIDI_REF
        LSR.L     #2,D1             ; E DIVIDILA PER 4
        BCLR      #0,D1             ; D1 PARI  PER  FORZA
        SUBQ.W    #1,D1             ; -1 PER DBRA=NUM. BYTES RIGA
        SUBQ.W    #1,D2             ; -1 PER DBRA=NUM. DELLE COLONNE
        MOVE.L    D1,D6             ; SALVA NUMERO BYTES IN D6
        TST.L     D5                ; CHE TIPO DI OVER ?
        BEQ.S     OVER0             ; OVER 0
        BPL.S     OVER1             ; OVER 1
        BMI.S     XOVER             ; OVER -1
OVER0
        MOVE.B    (A5)+,(A1)+       ; COPIA TUTTO
        DBRA      D1,OVER0          ; PER OGNI RIGA
        ADD.L     #128,A3           ; NUOVA RIGA
        MOVE.L    A3,A1             ; RIPRISTINA RIGA
        MOVE.W    D6,D1             ; RIPRISTINA LUNG RIGA

        DBRA      D2,OVER0          ; PER NUM COLONNE
        BRA.S     FINE              ; E RITORNA
OVER1
        MOVE.B    (A5)+,D4          ; COME OVER 0 MA CON OR
        OR.B      D4,(A1)+
        DBRA      D1,OVER1
        ADD.L     #128,A3
        MOVE.L    A3,A1
        MOVE.W    D6,D1
        DBRA      D2,OVER1
        BRA.S     FINE
XOVER
        MOVE.B    (A5)+,D4          ; COME OVER 0 MA CON XOR
        EOR.B     D4,(A1)+
```

```
        DBRA    D1,XOVER
        ADD.L   #128,A3
        MOVE.L  A3,A1
        MOVE.W  D6,D1
        DBRA    D2,XOVER
FINE
        MOVEQ   #0,D0               ; NESSUN ERRORE
        RTS                         ; RITORNA
SIZE
        CMPA.L  A3,A5               ; NESSUN. PARAMETRO ?
        BEQ     ALL_DEF_SIZE        ; TUTTO_DEFAULT=ALL_DEF_SIZE
        BTST    #7,1(A6,A3.L)       ; il primo parametro e' '#' ?
        BNE     DEF_SIZE            ; SI: GO TO  DEFault_SIZE
        BSR     PREL                ; GO SUB PRELeva parametri
        TST.L   D0                  ; ERRORI ?
        BNE.S   GO_BAS              ; SI: TORNALI AL BASIC
        CMPI.W  #2,D3               ; QUANTI PARAMETRI 4 ?
        BEQ.S   DUE_PARAMETRI       ; SI SALTA A DUE_PARAMETRI
        CMPI.W  #4,D3               ; QUANTI PARAMETRI 4 ?
        BEQ.S   QUATTRO_PARAMETRI   ; SI SALTA A QUATTRO PAR.
        BRA     BADPAR              ; NUM. PARAMETRI ERRATO
DUE_PARAMETRI
        MOVE.L  0(A6,A1.L),D1       ; LARGHEZZA
        MOVE.L  4(A6,A1.L),D2       ; ALTEZZA
        MOVEQ   #0,D3               ; AT 0,0
        MOVEQ   #0,D4
        BRA.S   CALCOLO             ; CALCOLA E RITORNA
QUATTRO_PARAMETRI
        MOVE.L   0(A6,A1.L),D1      ; LARGHEZZA
        MOVE.L   4(A6,A1.L),D2      ; ALTEZZA
        MOVE.L   8(A6,A1.L),D3      ; XPOS
        MOVE.L  12(A6,A1.L),D4      ; YPOS
CALCOLO
        BSR     CONTROLLA_PAR       ; CONTROLLA I PARAMETRI
        TST.L   D0                  ; SEGNALA ERRORI ?
        BNE.S   GO_BAS              ; SI TORNALI AL BASIC
        CMPI.W  #8,D1               ; CONTROLLA LARGhezza minima
        BGE.S   DIV_FUNZ
        MOVEQ   #9,D1
DIV_FUNZ
        LSR.L   #2,D1               ; D1=D1 DIV 4=NUM. BYTES LARG
        MULU    D2,D1               ; D1=D1*D2=BYTES LARG*ALT
        ADDQ.L  #8,D1               ; AGGIUNGI 8 BYTES  PER  L'HEADER
        MOVE.L  $58(A6),A1          ; SUBROUTINE LI_TO_FP
        SUBQ.L  #6,A1               ; VEDI VARI NUMERI DI
        MOVE.W  #$820,D2            ; "PERSONAL COMPUTER"

XX2
        SUBQ.W  #1,D2
        ASL.L   #1,D1
        BVC.S   XX2
        ROXR.L  #1,D1
        MOVE.W  D2,0(A6,A1.L)
        MOVE.L  D1,2(A6,A1.L)
        MOVE.L  A1,$58(A6)
```

```
        MOVEQ     #2,D4
        MOVEQ     #0,D0
GO_BAS
        RTS
FIND_CAN
        MOVE.W    $118,A4           ; $118 PRELEVA INTERI LUNGHI
        JSR       (A4)              ; CHIAMA CA_GTLIN
        TST.L     D0                ; ci sono ERRORI ?
        BNE.S     GO_BAS            ; SI: TORNALI AL BASIC
        CMP.W     #2,D3             ; CI SONO 2 PARAMETRI ?
        BNE       BADPAR            ; NO: BADPAR
        MOVE.L    0(A6,A1.L),D0     ; 1^ PARAM. in D0 (CANALE)
        MOVE.L    4(A6,A1.L),A5     ; 2^ PARAM. in A5 (BUFFER)
RETDEF
        CMP.L     #131072,A5        ; Buffer MINORE DI 131072 ?
        BLT       BADPAR            ; SI: BADPAR
IN_RANGE
        CMP.L     #255,D0           ; D0 e' maggiore 255?
        BGT       BADPAR            ; si: BADPAR
        BSR.S     GETCHAN           ; routine che cerca il CHAN.ID di D0
        TST.L     D0                ; segnala errori?
        BNE       FUORI             ; tornali al BASIC
        MOVE.L    A0,D1             ; D1=A0 (A0=CHAN.ID)
        MULU      #4,D1             ; D1=D1*4
        MOVE.L    D1,A1             ; A1=D1
        ADD.L     163960,A1         ; A1=D1+PEEK_L(163960)
        MOVE.L    (A1),A2           ; A2=PEEK_L(A1)
        MOVE.L    A2,A0
        ADD.L     #42,A2            ; d2=d2+42
        MOVE.L    (A2),D1           ; D1=PEEK_L(d2)
        TST.L     D1                ; D1 e' negativo?
        BLT       BADPAR            ; SI: BADPAR
        ADD.L     #$18,A0           ; PUNTA ALLA DEFW
        MOVE.W    (A0)+,D3          ; POSX
        MOVE.W    (A0)+,D4          ; POSY
        MOVE.W    (A0)+,D1          ; LARG
        MOVE.W    (A0),D2           ; ALT
        BSR       CONTROLLA_PAR     ; Controlla i dati
        TST.L     D0                ; ERRORI ?
        BNE       BADPAR            ; SI: ALLORA DAI BADPAR E RITORNA
        RTS

; GETCHAN : TROVA L' ID DEL CANALE SPECIFICATO IN D0
; E LO TORNA IN A0

GETCHAN
        MOVE.L    $30(A6),A0
        MULU      #$28,D0
        ADD.L     D0,A0
        CMP.L     $34(A6),A0
        BGE.S     NOT_OPEN
        MOVE.L    0(A6,A0.L),D0
        BLT.S     NOT_OPEN
        MOVE.L    D0,A0
        MOVEQ     #0,D0
```

```
        RTS
NOT_OPEN
        MOVEQ     #-6,D0            ; D0=-6: NOT OPEN
GO_OUT
        RTS                         ; RETurn (BASIC)
DEFAULT_CAN
        MOVE.L    0(A6,A1.L),A5     ; IN A5 BUFFER
        CMP.L     #131072,A5        ; BUFFER MINORE DI 131072 ?
        BLT       BADPAR            ; SI BADPAR
        MOVEQ     #1,D0             ; CANALE DI  DEFAULT=1
        BSR       RETDEF            ; GOTO  RETurnDEFault
        BRA       POST_INPUT        ; SALTA A POST_INPUT
DEF_SIZE
        MOVEA.W   CA_GTLIN,A2       ; PRELEVA LONG_INTEGER
        JSR       (A2)              ; ESEGUI !
        BNE.S     GO_OUT            ; ERRORI ? SI:AL BASIC
        CMPI.W    #1,D3             ; UN SOLO PARAMETRO ?
        BNE       BADPAR            ; NO TORNA CON ERR_BP
        MOVE.L    0(A6,A1.L),D0     ; PARAMETRO PASSATO IN D0
PREL_SIZE_PAR
        BSR       IN_RANGE          ; CONTROLLA DATI CANALE
        TST.L     D0                ; ERRORI ?
        BNE.S     GO_OUT            ; SI:TORNALI AL BASIC
        BRA       CALCOLO           ; SALTA A CALCOLO
ALL_DEF_SIZE
        MOVEQ     #1,D0             ; DEFAULT CANALE 1
        BRA.S     PREL_SIZE_PAR     ; SALTA A PREL_SIZE_PAR

;       END
```

```
100 :
110 REMark **********************
120 REMark *    ~ANIMATE_BAS~    *
130 REMark * DEMO NUOVI COMANDI  *
140 REMark *                     *
150 REMark * SCR_STORE           *
160 REMark * SCR_REFRESH         *
170 REMark * SCR_SIZE            *
180 REMark *                     *
190 REMark *  NOTARIANNI CARLO   *
200 REMark *  NOTARIANNI MAURO   *
210 REMark **********************
220 :
230 dummy=RESPR(900)
240 LBYTES mdv1_GRAF_CODE,dummy
250 CALL dummy
260 :
270 MODE 4:WINDOW 512,256,0,0
280 WINDOW#0,448,40,32,216
290 PAPER 0:INK 7:CLS
300 WINDOW 448,200,32,16
310 OPEN#3,SCR_130X100A160X80
320 INK#3,7:SCALE#3,120,70,45
330 DIM a(100,2),b(100):RESTORE 1050
340 cont=0:a1=0:a2=0:a3=0
350 Buffer=RESPR(34000):buff=Buffer
360 REMark Usare ALCHP (se possibile)
370 :
380 REPeat loop
390  IF EOF:EXIT loop
400  READ c:a1=a1+c
410  IF a1<a2:a2=a1
420  IF a1>a3:a3=a1
430  cont=cont+1
440  READ d:a(cont,1)=c:a(cont,2)=d
450 END REPeat loop
460 :
470 largh=a3-a2
480 prof=20:ved=9
490 pp=(127+largh/2)-a3
500 :
510 FOR n=0 TO PI/2 STEP PI/(2*ved)
520  FOR m=1 TO cont:b(m)=a(m,2)*COS(
n)
530  z=(INT(((PI/2-n)*(90/PI*2))*100+
.5))/100
540  CLS#3:POINT#3,pp,70
550  FOR m=1 TO cont:LINE_R#3 TO a(m,
1),b(m)
560  x=prof*SIN(n):POINT#3,pp,70
570  LINE_R#3 TO 0,-x
580  FOR m=1 TO cont:LINE_R#3 TO a(m,
1),b(m):LINE_R#3 TO 0,x:LINE_R#3 TO 0
,-x
590  SCR_STORE 132,102,160,80 TO buff
600  BEEP 800,10:buff=buff+3400
610 END FOR n
620 :
```

```
630 DO_ANIMATE
640 :
650 DEFine PROCedure DO_ANIMATE
660  WINDOW#3,512,216,0,0:CLS#3:CLS#0
670  AT#0,0,15:PRINT#0;"Usa i tasti c
ursore ()=?<) per muovere"
680  AT#0,1,15:PRINT#0;"la figura sul
lo schermo."
690  AT#0,3,8:PRINT#0;"ALT/CTRL/SHIFT
 aumentano la velocita' di spostament
o."
700  :
710  REMark *** animazione ***
720  :
730  Xp=160:Yp=80
740  WINDOW#3,132,102,Xp,Yp
750  REPeat show_loop
760   K=KEYROW(1)
770   IF K THEN
780    cont=0
790    REPeat MOVE_LOOP
800     K=KEYROW(1):ST=KEYROW(7)+1
810       cont=cont*(K=0):cont=cont+1
820       IF cont=50:EXIT MOVE_LOOP
830       IF NOT K :NEXT MOVE_LOOP
840       IF K&&2  :Xp=Xp-ST*(Xp>=ST)
850       IF K&&16 :Xp=Xp+ST*(Xp<=(51
2-132-ST))
860       IF K&&4  :Yp=Yp-ST*(Yp>=ST)
870       IF K&&128:Yp=Yp+ST*(Yp<=(21
6-102-ST))
880       IF K&&64 :CLS:EXIT MOVE_LOO
P
890       BORDER #3,1,0:CLS#3:WINDOW#
3,132,102,Xp,Yp:BORDER#3,1,7
900    END REPeat MOVE_LOOP
910    BORDER#3,1,0
920   END IF
930    :
940    FOR fig1=Buffer TO Buffer+3060
0 STEP 3400
950      SCR_REFRESH fig1,Xp,Yp
960    END FOR fig1
970    :
980    FOR fig2=Buffer+30600 TO Buffe
r STEP -3400
990      SCR_REFRESH fig2,Xp,Yp
1000    END FOR fig2
1010    :
1020  END REPeat show_loop
1030 END DEFine DO_ANIMATE
1040 :
1050 DATA -55,0,0,30,-15,-30,-15,7,-4
,-7,-22,0,20,40,15,-10,30,60,30,-60,-
20,0,-3,5,0,-8,25,0,14,-27
1060 :
1070 DEFine PROCedure WINDOW_SAVE (De
v_name$,Larg,Alt,Xpos,Ypos)
1080 LOCal Buffer,Lung
1090   Lung=SCR_SIZE(Larg,Alt)
1100   Buffer=ALCHP(Lung)
1110   SCR_STORE Larg,Alt,Xpos,Ypos T
O Buffer
1120   SBYTES Dev_name$,Buffer,Lung
1130   RECHP Buffer
1140 END DEFine WINDOW_SAVE
1150 :
1160 DEFine PROCedure CHANNEL_SAVE (C
an,Dev_name$)
1170 LOCal Buffer,Lung
1180   Lung=SCR_SIZE(#Can)
1190   Buffer=ALCHP(Lung)
1200   SCR_STORE#Can,Buffer
1210   SBYTES Dev_name$,Buffer,Lung
1220   RECHP Buffer
1230 END DEFine CHANNEL_SAVE
1240 :
1250 DEFine PROCedure WINDOW_LOAD (De
v_name$)
1260 LOCal Buffer,Lung
1270   OPEN_IN#3,Dev_name$
1280   Lung=FLEN(#3):CLOSE#3
1290   Buffer=ALCHP(Lung)
1300   LBYTES Dev_name$,Buffer
1310   SCR_REFRESH Buffer
1320   RECHP Buffer
1330 END DEFine WINDOW_LOAD
1340 :
```

GARE CON L'ARCO

```
100 REMark  BY FABIO CAVICCHIO 1987
110 HI=0
120 INIZIO
130 REPeat gioco
140 PUNTI1=0:PUNTI2=0
150 FOR A=45 TO 70 STEP 5:FRECCIA1 10
0,A:FRECCIA2 325,A:NEXT A
160 CLS#3:CSIZE#3,0,0:AT#3,0,13:INK#3
,7:PRINT#3,"PREMI 1 - 2  PER LA SCELT
A DEL NUMERO DEI GIOCATORI"
170 REPeat A
180 IF INKEY$="1" THEN GIOC =1:EXIT A
190 IF INKEY$="2" THEN GIOC =2:EXIT A
:ELSE :END REPeat A
200 CLS#3:INK#3,0,7,1:CSIZE#3,2,1:PRI
NT#3,"PREMI [SPACE] PER SCAGLIARE LE
FRECCE":FOR A=1 TO 10:PAUSE 20
210 CLS#3:PRINT#3," 00000 < I";:AT#3,
0,28:INK#3,0,4,1:PRINT#3,"II = 00000"
:INK#3,0,7,1
220 FOR V=1 TO GIOC*6
230 GARA :IF GIOC=2 AND V/2 = V DIV 2
 THEN INK#3,0,4,1:AT#3,0,16:PRINT#3,P
UNTI;"    ";PUNTI2=PUNTI2+PUNTI:AT#3,0
,38-LEN (PUNTI2):PRINT#3,PUNTI2:ELSE
:INK#3,0,7,1:AT#3,0,16:PRINT#3,PUNTI;
"    ":PUNTI1=PUNTI1+PUNTI:AT#3,0,6-LE
N(PUNTI1):PRINT#3,PUNTI1;
240 END FOR V
250 IF GIOC =1 THEN PUNTI=PUNTI1:GO T
O 280
260 IF PUNTI1<PUNTI2 THEN PUNTI=PUNTI
2:W=2:ELSE : PUNTI=PUNTI1:W=1
270 PAUSE 200:CLS#3:INK#3,4,7:IF PUNT
I1=PUNTI2 THEN PRINT#3,,"LA GARA E'FI
NITA IN PARITA'":ELSE :PRINT#3,"  IL
VINCITORE E'IL GIOCATORE N."!W
280 PAUSE 150:A$="RIMANE FISSATO A":I
```

```
F HI< PUNTI1 OR HI <PUNTI2 THEN A$="E
'ORA FISSATO A":HI=PUNTI
290 CLS#3:PRINT#3,"IL RECORD"!A$!HI!"
PUNTI"
300 PAUSE 200:CLS#3:TABELLONE
310 BEEP 30000,20,30,3000,7,1
320 BEEP 30000,20,30,3000,-8,1:END RE
Peat gioco
330 STOP
340 DEFine PROCedure FRECCIA1 (X,Y):B
LOCK 35,3,X,Y,0:BLOCK 34,1,X+1,Y+1,7:
BLOCK 3,5,X+36,Y-1,2:BLOCK 3,3,X+39,Y
,2:BLOCK 2,1,X+42,Y+1,2:BLOCK 2,3,X+3
7,Y,7:BLOCK 3,1,X+39,Y+1,7:END DEFine

350 DEFine PROCedure FRECCIA2 (X,Y):B
LOCK 35,3,X,Y,0:BLOCK 34,1,X+1,Y+1,4:
BLOCK 3,5,X+36,Y-1,0:BLOCK 3,3,X+39,Y
,0:BLOCK 2,1,X+42,Y+1,0:BLOCK 2,3,X+3
7,Y,7:BLOCK 3,1,X+39,Y+1,7:END DEFine

360 REMark
370 DEFine PROCedure TABELLONE
380 PAPER#4,0,2:CLS#4
390 INK#4,0:FILL#4,1:CIRCLE#4,50,50,5
0:FILL#4,0:INK#4,2:CIRCLE#4,50,50,36:
CIRCLE#4,50,50,50:FILL#4,1:CIRCLE#4,5
0,50,36:INK#4,4:FILL#4,0:CIRCLE#4,50,
50,22:FILL#4,1:CIRCLE#4,50,50,22:FILL
#4,0:INK#4,7:CIRCLE#4,50,50,8:FILL#4,
1:CIRCLE#4,50,50,8:FILL#4,0
400 END DEFine
410 REMark
420 DEFine PROCedure MOV:WINDOW 427,5
,21,147:END DEFine
430 DEFine PROCedure NORM:WINDOW 466,
256,21,0:END DEFine
440 REMark
+1)*5,4:ELSE :FRECCIA1 20,148:BLOCK 4
4,5,100,34+((V+GIOC-1)/GIOC+1)*5,4
920 REPeat A
930 TASTIERA:IF FINE=1 THEN EXIT A
940 SCROLL#5,1:YY=YY+1:END REPeat A
950 BEEP TEMPO*300+5000,10,200,TEMPO*
2+5,2,1,1,3:STRIP 7:AT 19,18:PRINT TE
MPO;":":MOV:FOR A=1 TO 91:PAN 4:SCROL
L #5,1:YY=YY+1
960 IF PUNTI_X=0 THEN FOR A=1 TO 84:P
AN#7,1:NEXT A:ELSE :BEEP 200,15,230,2
0000,7,15,2,15
970 PUNTEGGIO:SEGNA:CLS:CLS#5:NORM:EN
D DEFine

450 DEFine PROCedure ERBA:INK 2:FOR R
=32 TO 255 STEP 16
460 FOR R1=0 TO 365 STEP 20
470 R2=RND(0 TO 7):R3=RND(0 TO 7):R2=
R2+R:R3=R3+R1:POINT R3,R2:END FOR R1
480 END FOR R
490 INK 0:END DEFine
500 REMark
510 DEFine PROCedure INIZIO:WINDOW 1,
512,256,0,0
520 POKE 98683,3:MODE 4:POKE 98683,3:
NORM
530 SCALE 255,0,0
540 OPEN #4,SCR_108X80A200X10:OPEN #8
,scr_512x256a0x0

550 OPEN #5,SCR_8X236A447X0:OPEN #7,S
CR_84X5A400X147
560 PAPER#5,4:PAPER#7,4
570 CSIZE#4,2,0:CSIZE 2,0
580  PAPER 5:INK 0:OPEN#3,SCR_466X20A
21X236:PAPER#3,0:INK#3,7
590 PAPER#8,0:CLS#8:CLS:CLS#3:ERBA:BL
OCK#8,108,80,206,14,0:TABELLONE:POKE
98683,0
600 END DEFine
610 REMark
620 DEFine PROCedure BERSAGLIO (YY):B
LOCK#5,2,22,5,YY,0:BLOCK#5,2,16,3,YY+
3,2:BLOCK#5,2,4,1,YY+9,7:END DEFine
630 REMark
640 DEFine FuNction PUNTI_X:P=0:SELec
t YY
650 ON YY=136 TO 139:P=20:ON YY=133 T
O 143:P=10:ON YY=130 TO 146:P=5:ON YY
=127 TO 149:P=1
660 END SELect :RETurn P:END DEFine
670 REMark
680 DEFine FuNction PUNTI_Y:P2=0:SELe
ct TEMPO
690 ON TEMPO=15:P2=20:ON TEMPO=14 TO
16:P2=18:ON TEMPO=13 TO 17:P2=15:ON T
EMPO=12 TO 18:P2=12:ON TEMPO=11 TO 19
:P2=8:ON TEMPO=10 TO 20:P2=1
700 END SELect :RETurn P2:END DEFine
710 REMark
720 DEFine PROCedure TASTIERA
730 IF FT=1 THEN GO TO 750
740 IF KEYROW(1)=64 THEN FT=1:GO TO 7
50:ELSE :END DEFine
750 TEMPO=TEMPO+1:IF KEYROW(1)=64 THE
N END DEFine
760 FINE=1
770 END DEFine
780 REMark
790 DEFine PROCedure PUNTEGGIO:PUNTI=
PUNTI_Y*PUNTI_X*10:IF PUNTI=10 THEN P
UNTI =0
800 END DEFine
810 REMark
820 DEFine PROCedure SEGNA:IF PUNTI=0
 THEN END DEFine
830 CY=((20-TEMPO) *6.7)+RND(-1 TO 1)
+2
840 A=149-YY
850 CX=(A/22*91)+RND(-2 TO 2)
860 IF CX <0 OR CY<0 THEN GO TO 880
870 INK#4,0:OVER#4,1:CURSOR#4,CX,CY:P
RINT#4,"X":FOR A=2 TO 7+V MOD 2-GIOC
STEP 2:INK#4,A:CURSOR#4, CX+1,CY-1:PR
INT#4,"X":END FOR A:OVER#4,0
880 END DEFine
890 REMark
900 DEFine PROCedure GARA
910 TEMPO=0:FT=0:FINE=0:YY=0:BERSAGLI
O YY:IF GIOC=2 AND V/2=V DIV 2 THEN F
RECCIA2 20,148:BLOCK 44,5,325,34+(V/2
```

OTHELLO

```
100 REM ************************
110 REM ******* OTHELLO *******
120 REM * di Matteo Bertolini *
```

```
 130 REM ** versione Spectrum **
 140 REM **********************
 150 REM
 200 PAPER 4: BORDER 4
 210 CLS
 220 RANDOMIZE
 300 PRINT AT 0,12; PAPER 4;"OTH
ELLO"
 310 PRINT AT 10,2; PAPER 4;"Vuo
i giocare contro di me ?"
 315 INPUT c$
 320 IF c$="s" OR c$="S" THEN  L
ET comp=1
 325 IF c$="n" OR c$="S" THEN  L
ET comp=0
 500 BORDER 5
 550 CLS
 597 REM
 598 REM DISEGNA IL RETTANGOLO
 599 REM
 605 DRAW 166,0
 610 DRAW 0,166
 620 DRAW -166,0
 630 DRAW 0,-166
 640 FOR c=0 TO 166 STEP 20.75
 650 PLOT 0,c: DRAW 166,0
 660 PLOT c,0: DRAW 0,166
 670 NEXT c
 675 PRINT AT 0,0;" 1  2 3  4 5
 6 7  8"
 680 PRINT AT 2,21;"8";AT 5,21;"
7";AT 7,21;"6";AT 10,21;"5";AT 1
2,21;"4";AT 15,21;"3";AT 17,21;"
2";AT 20,21;"1"
 685 FOR a=0 TO 21
 690 PRINT AT a,23; PAPER 6;"
  "
 695 NEXT a
 700 CIRCLE 20.5*3+11,20.5*3+11,
6: CIRCLE 20.5*4+11,20.5*4+11,6
 710 CIRCLE 20.5*3+11,20.5*4+11,
6: CIRCLE 20.5*3+11,20.5*4+11,3:
 CIRCLE 20.5*4+11,20.5*3+11,6: C
IRCLE 20.5*4+11,20.5*3+11,3
 997 REM
 998 REM DEFINIZIONE VARIABILI
 999 REM
1000 DIM a(10,10)
1005 DIM b(64,3)
1010 LET a(5,5)=1: LET a(6,6)=1:
 LET a(5,6)=-1: LET a(6,5)=-1
1100 LET gioc=1
1110 LET g=1
1115 LET mosse=0: LET mossef=0
1120 LET ped=0
1130 LET nc=0
1210 GO SUB 9800
1220 LET contr=1300
1225 LET cm=0
1310 REM
1320 REM IMMISSIONE DATI
1330 REM
1332 IF mosse=60 THEN  GO SUB 90
00: STOP
1335 LET mosse=mosse+1
1336 PRINT AT 5,24; PAPER 6;"Mos
sa ";mosse
1338 IF comp=1 AND g=2 THEN  LET
 contr=9500: PRINT AT 2,24; PAPE
R 6;"COMPUTER";AT 3,24;"
": GO TO 2000
1340 PRINT AT 2,24; PAPER 6;"GIO
C N";g;" "
1350 PRINT AT 3,24; PAPER 6;"Ins
. x,y"
1360 INPUT  LINE d$
1362 IF d$="m" THEN  GO SUB 9850
1363 IF d$="l" THEN  GO SUB 9900
: GO TO 1336
1364 IF d$="c" THEN  COPY
1365 IF d$="p" THEN  LET contr=9
500: GO TO 2000
1367 IF d$="s" THEN  GO SUB 9020
: PAUSE 100: PRINT AT 15,24; PAP
ER 6;"          ";AT 16,24;"
  ": GO TO 1340
1368 IF LEN (d$)<>3 THEN  GO TO
1360
1370 LET x=VAL (d$(1))+1
1380 LET y=VAL (d$(3))+1
1390 PRINT AT 2,24; PAPER 6;"
    ";AT 3,24;"          "
1997 REM
1998 REM CONTROLLI
1999 REM
2000 IF contr=9500 THEN  FOR x=2
 TO 9: FOR y=2 TO 9
2003 IF x<2 OR x>9 THEN  GO SUB
8000: GO TO contr
2005 IF y<2 OR y>9 THEN  GO SUB
8000: GO TO contr
2010 IF a(x,y)<>0 THEN  GO SUB 8
000: GO TO contr
2020 IF a(x+1,y+1)<>gioc*-1 AND
a(x,y+1)<>gioc*-1 AND a(x-1,y+1)
<>gioc*-1 AND a(x+1,y)<>gioc*-1
AND a(x-1,y)<>gioc*-1 AND a(x+1,

y-1)<>gioc*-1 AND a(x,y-1)<>gioc
*-1 AND a(x-1,y-1)<>gioc*-1 THEN
  GO SUB 8000: GO TO contr
2030 FOR z=x+1 TO 9
2035 IF a(z,y)=0 THEN  LET z=9:
NEXT z
2040 IF a(z,y)=gioc AND a(x+1,y)
=gioc*-1 THEN  LET ped=ped+(z-x-
1): LET ia=3000: LET z=9: NEXT z
2050 NEXT z
2060 FOR z=x-1 TO 2 STEP -1
2065 IF a(z,y)=0 THEN  LET z=2:
NEXT z
2070 IF a(z,y)=gioc AND a(x-1,y)
=gioc*-1 THEN  LET ped=ped+(x-z-
1): LET ib=3500: LET z=2: NEXT z
2080 NEXT z
2100 FOR t=y+1 TO 9
2105 IF a(x,t)=0 THEN  LET t=9:
NEXT t
2110 IF a(x,t)=gioc AND a(x,y+1)
=gioc*-1 THEN  LET ped=ped+(t-y-
1): LET ic=4000: LET t=9: NEXT t
2120 NEXT t
2130 FOR t=y-1 TO 2 STEP -1
2135 IF a(x,t)=0 THEN  LET t=2:
NEXT t
2140 IF a(x,t)=gioc AND a(x,y-1)
=gioc*-1 THEN  LET ped=ped+(y-t-
1): LET id=4500: LET t=2: NEXT t
2150 NEXT t
2197 REM

2198 REM CONTROLLI DIAGONALI
2199 REM
2200 LET t=y
2210 FOR z=x+1 TO 9
2220 LET t=t+1
```

```
2225 IF a(z,t)=0 THEN  LET z=9:
NEXT z
2230 IF a(z,t)=gioc AND a(x+1,y+
1)=gioc*-1 THEN  LET ped=ped+(z-
x-1): LET ie=5000: LET z=9: NEXT
 z
2235 IF t=9 THEN  LET z=9
2240 NEXT z
2250 LET t=y
2260 FOR z=x-1 TO 2 STEP -1
2270 LET t=t+1
2275 IF a(z,t)=0 THEN  LET z=2:
NEXT z
2280 IF a(z,t)=gioc AND a(x-1,y+
1)=gioc*-1 THEN  LET ped=ped+(x-
z-1): LET ig=5500: LET z=2: NEXT
 z
2285 IF t=9 THEN  LET z=2
2290 NEXT z
2300 LET t=y
2310 FOR z=x+1 TO 9
2320 LET t=t-1
2325 IF a(z,t)=0 THEN  LET z=9:
NEXT z
2330 IF a(z,t)=gioc AND a(x+1,y-
1)=gioc*-1 THEN  LET ped=ped+(z-
x-1): LET ih=6000: LET z=9: NEXT
 z
2335 IF t=2 THEN  LET z=9
2340 NEXT z
2350 LET t=y
2360 FOR z=x-1 TO 2 STEP -1
2370 LET t=t-1
2375 IF a(z,t)=0 THEN  LET z=2:
NEXT z
2380 IF a(z,t)=gioc AND a(x-1,y-
1)=gioc*-1 THEN  LET ped=ped+(x-
z-1): LET ii=6500: LET z=2: NEXT
 z
2385 IF t=2 THEN  LET z=2
2390 NEXT z
2392 IF contr=9500 AND ped<>0 AN
D comp=1 AND g=2 THEN  GO SUB 85
00: GO TO 9500
2395 IF ped=0 THEN  GO SUB 8000:
 GO TO contr
2396 IF contr=9500 AND ped<>0 TH
EN  PRINT AT 10,24; PAPER 6;"MUO
VI!": PAUSE 100: PRINT AT 10,24;
 PAPER 6;"        ": LET mosse=mo
sse-1: GO SUB 9800: GO TO 1220
2397 REM

2398 REM STAMPA PEDINA
2399 REM
2400 IF g=1 THEN  LET a=1
2405 IF g=2 THEN  LET a=0
2410 CIRCLE 20.5*(x-2)+11,20.5*(
y-2)+11,6: CIRCLE  INVERSE a;20.
5*(x-2)+11,20.5*(y-2)+11,3
2420 LET a(x,y)=gioc
2450 GO SUB ia
2452 GO SUB ib
2454 GO SUB ic
2456 GO SUB id
2458 GO SUB ie
2460 GO SUB ig
2462 GO SUB ih
2464 GO SUB ii
2597 REM

2598 REM CAMBIO GIOCATORE
2599 REM
2600 IF gioc=1 THEN  LET gioc=-1
: GO TO 2610
2605 IF gioc=-1 THEN  LET gioc=1
2610 IF g=1 THEN  LET g=2: GO TO
 2620
2615 IF g=2 THEN  LET g=1
2620 GO TO 1120
2630 STOP
2997 REM

2998 REM ROUTINES CAMBIO PEDINE
2999 REM
3000 REM X+1
3005 LET xn=x: LET yn=y
3020 LET xn=xn+1
3030 IF a(xn,y)=gioc THEN  RETUR
N
3040 GO SUB 6600

3060 GO TO 3020
3500 REM x-1
3505 LET xn=x: LET yn=y
3520 LET xn=xn-1
3530 IF a(xn,y)=gioc THEN  RETUR
N
3540 GO SUB 6600
3560 GO TO 3520

4000 REM y+1
4005 LET yn=y: LET xn=x
4010 LET yn=yn+1
4020 IF a(x,yn)=gioc THEN  RETUR
N
4030 GO SUB 6600
4050 GO TO 4010

4500 REM y-1
4505 LET yn=y: LET xn=x
4510 LET yn=yn-1
4520 IF a(x,yn)=gioc THEN  RETUR
N
4530 GO SUB 6600

4550 GO TO 4510
5000 REM X+1,Y+1
5010 LET yn=y: LET xn=x
5020 LET xn=xn+1: LET yn=yn+1
5030 IF a(xn,yn)=gioc THEN  RETU
RN
5040 GO SUB 6600
5060 GO TO 5020
5500 REM X-1,Y+1
5510 LET yn=y: LET xn=x
5520 LET yn=yn+1: LET xn=xn-1
5530 IF a(xn,yn)=gioc THEN  RETU
RN
5540 GO SUB 6600
5560 GO TO 5520
6000 REM X+1,y-1
6010 LET yn=y: LET xn=x
6020 LET xn=xn+1: LET yn=yn-1
6030 IF a(xn,yn)=gioc THEN  RETU
RN

6040 GO SUB 6600
6060 GO TO 6020
6500 REM X-1,Y-1
6510 LET yn=y: LET xn=x
6520 LET xn=xn-1: LET yn=yn-1
6530 IF a(xn,yn)=gioc THEN  RETU
RN
6540 GO SUB 6600
6560 GO TO 6520
6597 REM
```

```
6598 REM STAMPA PEDINE
6599 REM
6600 LET a(xn,yn)=gioc
6610 CIRCLE 20.5*(xn-2)+11,20.5*
(yn-2)+11,6: CIRCLE  INVERSE a;2
0.5*(xn-2)+11,20.5*(yn-2)+11,3
6620 RETURN
6797 REM

6798 REM  GO SUB nullo
6799 REM
6800 RETURN
7997 REM
7998 REM ROUTINE DI ERRORE

7999 REM
8000 IF contr=9500 THEN  RETURN
8002 IF g=2 AND comp=1 THEN  LET
 nc=1: LET contr=9500: PRINT AT
10,24; PAPER 6;"PASSO": PAUSE 10
0: PRINT AT 10,24; PAPER 6;"
 ": RETURN
8005 PRINT AT 10,24; PAPER 6;"ER
RORE!"
8010 PAUSE 100
8015 PRINT AT 10,24; PAPER 6;"
      "
8020 LET mosse=mosse-1
8030 RETURN
8497 REM

8498 REM STABILISCE PRIORITA'
8499 REM
8500 LET cm=cm+1
8510 LET b(cm,2)=x
8520 LET b(cm,3)=y
8530 IF x=3 OR x=8 OR y=3 OR y=8
 THEN  LET ped=ped-50
8540 IF (x=3 AND y=3) OR (x=3 AN
D y=8) OR (x=8 AND y=3) OR (x=8
AND y=8) THEN  LET ped=ped-50
8550 IF x=2 OR x=9 OR y=2 OR y=9
 THEN  LET ped=ped+50
8560 IF (x=2 AND y=2) OR (x=2 AN
D y=9) OR (x=9 AND y=2) OR (x=9
AND y=9) THEN  LET ped=ped+50
8568 IF (x=2 AND y=3) OR (x=2 AN
D y=8) OR (x=8 AND y=2) OR (x=8
AND y=9) OR (x=3 AND y=2) OR (x=
3 AND y=9) OR (x=9 AND y=3) OR (
x=9 AND y=8) THEN  LET ped=ped-1
00
8570 LET ped=ped+RND
8575 LET b(cm,1)=ped
8578 GO SUB 9800
8580 RETURN
8597 REM

8598 REM STABILISCE MOSSA
8599 REM
8600 FOR c=1 TO cm
8610 IF b(1,1)<b(c,1) THEN  FOR
d=1 TO 3: LET b(1,d)=b(c,d): NEX
T d
8620 NEXT c
8625 LET x=b(1,2): LET y=b(1,3)
8630 RETURN
8997 REM

8998 REM FINE GIOCO
8999 REM
9000 PRINT AT 10,24; PAPER 6;"FI
NE";AT 11,24;"PARTITA"
9010 PRINT AT 13,24; PAPER 6;"SI
T.FIN."
9020 LET bianco=0: LET nero=0
9025 FOR X=2 TO 9
9030 FOR Y=2 TO 9
9040 IF a(X,Y)=1 THEN  LET bianc
o=bianco+1
9045 IF a(X,Y)=-1 THEN  LET nero
=nero+1
9050 PRINT AT 15,24; PAPER 6;"Bi
anco";bianco
9060 PRINT AT 16,24; PAPER 6;"Ne
ro  ";nero
9070 NEXT y
9080 NEXT x
9090 RETURN
9497 REM

9498 REM ROUTINE DI PASSO MOSSA
9499 REM
9500 LET ped=0
9502 NEXT y: NEXT x
9505 IF comp=1 AND g=2 AND nc=0
THEN  GO SUB 8600: LET contr=130
0: GO TO 2000
9509 LET mosse=mosse-1

9511 IF mossef=mosse THEN  PRINT
 AT 2,24; PAPER 6;"          ";AT
3,24;"          ": GO SUB 9000: ST
OP
9512 LET mossef=mosse

9513 IF gioc=1 THEN  LET gioc=-1
: GO TO 9515
9514 IF gioc=-1 THEN  LET gioc=1
9515 IF g=1 THEN  LET g=2: GO TO
 9520
9516 IF g=2 THEN  LET g=1
9520 GO SUB 9800: GO TO 1120
9797 REM

9798 REM AZZERA I CAMBI PEDINA
9799 REM
9800 LET ia=6800: LET ib=6800: L

ET ic=6800: LET id=6800: LET ie=
6800: LET ig=6800: LET ih=6800:
LET ii=6800
9810 RETURN
9847 REM

9848 REM MEMORIZZA PARTITA
9849 REM
9850 PRINT AT 8,24; PAPER 6;"Nom
e ?"
9852 INPUT n$
9853 IF LEN n$>8 THEN  GO TO 985
0
9854 LET a(10,10)=mosse: LET a(9
,10)=g: LET a(8,10)=gioc
9855 PRINT AT 8,24; PAPER 6;"Tap
e o";AT 9,24;"Disk ?"
9860 IF INKEY$<>"t" AND INKEY$<>
"d" THEN  GO TO 9860
9862 IF INKEY$="t" THEN  SAVE n$
 DATA a()
9864 IF INKEY$="d" THEN  REM mem
orizza da microdrive
9866 PRINT AT 8,24; PAPER 6;"
    ";AT 9,24;"      "
9867 LET a(8,10)=0: LET a(9,10)=
0: LET a(10,10)=0
9868 RETURN
```

```
9897 REM
9898 REM CARICA PARTITA
9899 REM

9900 PRINT AT 8,24; PAPER 6;"Nom
e ?"
9905 INPUT n$
9910 PRINT AT 8,24; PAPER 6;"Tap
e o";AT 9,24;"Disk ?"
9915 IF INKEY$<>"t" AND INKEY$<>
"d" THEN  GO TO 9915

9920 IF INKEY$="t" THEN  LOAD n$
 DATA a()
9925 IF INKEY$="d" THEN  REM car
ica da microdrive

9930 PRINT AT 8,24; PAPER 6;"
    ";AT 9,24;"       "
9935 LET mosse=a(10,10): LET a(1
0,10)=0
9936 LET g=a(9,10): LET a(9,10)=
0
9937 LET gioc=a(8,10): LET a(8,1
0)=0
9938 CLS

9939 DRAW 166,0: DRAW 0,166: DRA
W -166,0: DRAW 0,-166
9940 FOR c=0 TO 166 STEP 20.75:
PLOT 0,c: DRAW 166,0: PLOT c,0:
DRAW 0,166: NEXT c
9941 PRINT AT 0,0;"1  2 3  4 5
6 7  8"

9942 PRINT AT 2,21;"8";AT 5,21;"
7";AT 7,21;"6";AT 10,21;"5";AT 1
2,21;"4";AT 15,21;"3";AT 17,21;"
2";AT 20,21;"1"

9943 FOR a=0 TO 21: PRINT AT a,2
3; PAPER 6;"          ": NEXT a
9944 FOR x=2 TO 9
9945 FOR y=2 TO 9
9947 IF a(x,y)=0 THEN  GO TO 996
2
9950 IF a(x,y)=1 THEN  LET a=1
9955 IF a(x,y)=-1 THEN  LET a=0

9960 CIRCLE 20.5*(x-2)+11,20.5*(
y-2)+11,6: CIRCLE  INVERSE a;20.
5*(x-2)+11,20.5*(y-2)+11,3
9962 NEXT y
9964 NEXT x
9970 RETURN
```

DELETE FILES

```
 100 REM **********************
 101 REM *        win$        *
 102 REM **********************
 103 REM *   1987 E.Boscani   *
 104 REM **********************
 105 REM * GENERATORE BASIC  *
 106 REM **********************
 110 PAPER 0: INK 5: BORDER 0: 0
VER 0: BRIGHT 0: FLASH 0: POKE 2
3624,5
 120 POKE 23609,22: CLEAR 50499

 130 LET err=0: LET a=10: LET b=
11: LET c=12: LET d=13: LET e=14
: LET f=15
 140 LET l=9000: LET ad=64500

 150 RESTORE l: LET ck=0
 160 POKE 23692,99: READ a$,s$
 170 LET by=16*VAL a$(1)+VAL a$(
2)
 180 LET ck=ck+by: POKE ad,by: L
ET a$=a$(3 TO )
 200 LET ad=ad+1: IF a$<>"" THEN
  GO TO 170
 210 IF ck=VAL s$ THEN  PRINT TA
B 9;"Linea ";l;" OK!": LET l=l+1
: IF l<=9025 THEN  GO TO 150

 220 IF ck<>VAL s$ THEN  PRINT T
AB 9;"Linea ";l;" ERRATA": LET e
rr=1: LET l=l+1: IF l<=9025 THEN
  GO TO 150
 225 IF err THEN  PRINT '''"Fine
trasferimento codici"'"Ricontrol
la le linee errate": STOP
 230 CLS : PRINT "Salvataggio co
dici win$..."

 240 SAVE "win$"CODE 64500,1000
 250 PRINT '''"Salvataggio genera
tore BASIC..."
 260 SAVE "win$genbas"
 270 STOP : STOP : STOP : STOP

9000 DATA "1809000000000000000000
03E01DD21F6FB1104003 7CD56053AF7F
B4F3AF9FB8132FDFB4F","3174"
9001 DATA "3E1891878787473AF6FB3
2FCFB87878732FEFB4FDD2190FBDD22F
AFB1100003AF9FB8787","5104"
9002 DATA "87F5C53AF8FBF5C5D53EB
FCDAC22D1ED73B05CED7BFAFBE5ED73F
AFBED7BB05C13C1F13D","6367"
9003 DATA "B72808F579C6084FF118D
7C1F13DB7280904F53AFEFB4FF118C3D
53AFCFBCD282D3E20CD","4803"
9004 DATA "282D3AFDFB3DCD282DEF0
40F38CDA22D21005809D13AF8FBF5E53
AF9FBED73B05CED7BFA","4882"
9005 DATA "FBE5ED73FAFBED7BB05C1
3F5D5112000A7ED52D1F13DB720E0E1F
13DB728032318D2ED73","5297"
9006 DATA "B05CED7BFAFBDDE13EFF3
7140815F33E0FD3FEDBFE1FE620F6024
FBFC269FDCDE70530F8","5354"
9007 DATA "21150410FE2B7CB520F9C
DE30530E9069CCDE30530E23EC6B830E
02420F106C9CDE70530","4269"
9008 DATA "D378FED430F4CDE705D26
9FD79EE034F260006B0182308200730 1
1DD75001813CB11ADC2","3898"
9009 DATA "69FD791F4F131809DD7E0
0ADC269FDDDE11B0806B22E01CDE305D
269FD3ECBB8CB1506B0","4285"
9010 DATA "D24FFD7CAD677AB320C47
CFE01ED7BB05CFB3A485C1F1F1FD3FEC
900000000000000000","3710"
9011 DATA "000000000000000000000
000000000000000000000000000000000
000000000000000000","0"
9012 DATA "000000000000000000000
000000000000000000000000000000000
000000000000000000","0"
9013 DATA "000000000000000000000
000000000000000000000000000000000
```

```
ØØØØØØØØØØØ18Ø9ØØØØ","33"
9Ø14 DATA "Ø5182ØF3ØØ18ØØ2A655CE
D5B635CA7ED527DFE143ØØ2CF19CD991
E7932EDFDCD991E7932","3857"
9Ø15 DATA "ECFDCD991E7932EBFDCD9
91E7932EAFD32FØFD878787F53AEBFD4
F3AEDFD8132F1FD4F3E","5762"
9Ø16 DATA "189187878747F14F32F2F
DDD219ØFBDD22EEFD11ØØØØ3AEDFD878
787F5C53AECFDF5C5D5","5519"
9Ø17 DATA "3EBFCDAC22D1ED73BØ5CE
D7BEEFDE5ED73EEFDED7BBØ5C13C1F13
DB728Ø8F579C6Ø84FF1","5681"
9Ø18 DATA "18D7C1F13DB728Ø9Ø4F53
AF2FD4FF118C3D53AFØFDCD282D3E2ØC
D282D3AF1FD3DCD282D","4648"
9Ø19 DATA "EFØ4ØF38CDA22D21ØØ58Ø
9D13AECFDF5E53AEDFDED73BØ5CED7BE
EFDE5ED73EEFDED7BBØ","5713"
9Ø2Ø DATA "5C13F5D5112ØØØED52D1F
13DB72ØE1E1F13DB728Ø32318D3DD21E
AFDD511Ø4ØØ3EØ1CDC2","4348"
9Ø21 DATA "Ø4Ø164ØØCD3D1F3EFFD11
3ED73BØ5CED7BEEFD2198ØCØ8F33EØ24
71ØFED3FEEEØFØ6A42D","42Ø4"
9Ø22 DATA "2ØF5Ø525F2Ø3FFØ62F1ØF
ED3FE3EØDØ6371ØFED3FEØ1ØE3BØ86FC
332FF7AB328ØCDD6EØØ","3855"
9Ø23 DATA "7CAD673EØ137C35ØFF6C1
8F479CB781ØFE3ØØ4Ø6421ØFED3FEØ63
E2ØEFØ5AF3CCB15C23F","4Ø57"
9Ø24 DATA "FF1BDDE1Ø6313E7FDBFE1
F3ØØ77A3CC229FFØ63B1ØFEED7BBØ5CF
B3A485C1F1F1FD3FECD","44Ø2"
9Ø25 DATA "282DCD282DCD282DCD282
DC9ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
ØØØØØØØØØØØØØØØØØØØ","1156"
```

```
   1 REM    WIN$

      di Ezio Boscani

     Programma principale

  1Ø BORDER Ø: PAPER Ø: INK 7
  15 LET org=Ø: LET scr=Ø: LET x
=Ø: LET y=Ø: LET lx=Ø: LET ly=Ø

  2Ø CLS : PAPER 5: INK 1: LET a
$="Load / Save finestre video":
LET px=1: GO SUB 8ØØØ
  3Ø INVERSE 1: LET px=3: LET a$
="MENU": GO SUB 8ØØØ

  4Ø PRINT '"1 - Load SCREEN$ "
  5Ø PRINT '"2 - Set windows"
  6Ø PRINT '"3 - Save window"
  65 PRINT '"4 - Load window"

  7Ø LET px=13: PAPER 4: INK Ø:
LET a$="Status report": GO SUB 8
ØØØ
  8Ø LET a$="Schermo "+("non " A
ND scr=Ø)+"in memoria": LET px=1
5: INK 5: GO SUB 8ØØØ
  9Ø IF org=Ø OR scr=Ø THEN  GO
TO 12Ø
 1ØØ LET px=18: LET a$="Parametr
i per il salvataggio": PAPER 2:
INK 7: BRIGHT 1: GO SUB 8ØØØ: LE
T px=19
 11Ø PRINT "X=";x,"Y=";y,"LX=";l
y+1,"LY=";lx+1;
 115 PRINT #Ø;"Fai la tua scelta
"
 12Ø LET a$=INKEY$
 13Ø IF a$<"1" OR a$>"4" THEN  G
O TO 12Ø
 135 IF a$="4" THEN  CLS : PRINT
 #Ø;"Start tape...": RANDOMIZE U
SR 645ØØ: INPUT "": PRINT #Ø;"Pr
ess any key": PAUSE Ø
 14Ø IF a$="1" THEN  GO SUB 1ØØØ
 15Ø IF a$="2" THEN  GO SUB 2ØØØ
 16Ø IF a$="3" THEN  GO SUB 3ØØØ
 17Ø GO TO 2Ø
1ØØØ CLS : INPUT "Nome file ";a$
1ØØ5 POKE 23658,Ø
1Ø1Ø INPUT "Nastro/Mdrive ";m$
1Ø2Ø IF m$<>"m" AND m$<>"n" THEN
  GO TO 1Ø1Ø
1Ø3Ø IF m$="n" THEN  LET scr=1:
LOAD a$SCREEN$ : GO SUB 9Ø4Ø: RE
TURN
1Ø4Ø IF LEN a$ THEN  LET scr=1:
LOAD *"m";1;a$SCREEN$ : GO SUB 9
Ø4Ø: RETURN
1Ø5Ø BEEP .3,-2Ø: PRINT #Ø;AT Ø,
Ø;"Invalid name": PAUSE 5Ø: INPU
T "Nome file ";a$: GO TO 1Ø4Ø
2ØØØ CLS : LET l=27: POKE 49958,
l: LET org=Ø: IF NOT scr THEN  P
RINT #Ø;AT Ø,Ø;"Screen not prese
nt": PAUSE 1ØØ: RETURN
2ØØ5 LET x=Ø: LET y=x: LET lx=x:
 LET ly=x: GO SUB 9Ø1Ø
2Ø2Ø GO SUB 5ØØØ
2Ø3Ø LET a$=INKEY$
2Ø4Ø IF a$="Ø" THEN  IF org=Ø TH
EN  GO SUB 4Ø4Ø: LET org=1
2Ø5Ø IF a$>"4" AND a$<"9" THEN
GO SUB 4ØØØ
2Ø6Ø IF a$=CHR$ 13 THEN  RETURN
2Ø7Ø IF a$=CHR$ 7 THEN  GO SUB 2
1ØØ
2Ø8Ø GO TO 2Ø3Ø
21ØØ IF l=27 THEN  LET l=24: POK
E 49958,l: CLS : GO SUB 9Ø1Ø: GO
 SUB 22ØØ: RETURN
211Ø LET l=27: POKE 49958,l: GO
SUB 9Ø1Ø: GO SUB 22ØØ: RETURN
22ØØ IF org=Ø AND lx=Ø AND ly=Ø
THEN  GO TO 224Ø
221Ø IF x>21 THEN  GO SUB 5ØØ1
222Ø IF x<22 THEN  GO SUB 5ØØØ
224Ø IF x+lx<22 THEN  GO SUB 5ØØ
Ø
225Ø IF x+lx>21 THEN  GO SUB 5ØØ
1
226Ø RETURN
3ØØØ CLS : IF org=Ø THEN  PRINT
#Ø;AT Ø,Ø;"Parameters unsetted":
 PAUSE 1ØØ: RETURN
3ØØ5 IF INKEY$<>"" THEN  GO TO 3
ØØ5
3Ø1Ø CLS : LET a$="Fai partire i
```

```
l nastro": LET px=10: GO SUB 800
0: LET a$="e premi un tasto": LE
T px=11: GO SUB 8000: PAUSE 0
3015 LET lx=lx+1: LET ly=ly+1
3020 POKE 49958,27: GO SUB 9010:
 RANDOMIZE y OR x AND ly+lx*USR
65000
3025 LET lx=lx-1: LET ly=ly-1
3030 INPUT "": PRINT #0;AT 0,0;"
Salvata window": PAUSE 100
3040 RETURN
4000 IF x+lx<22 THEN  GO SUB 500
0
4001 IF x+lx>21 THEN  GO SUB 500
1
4010 IF org=1 THEN  GO TO 4200
4020 LET x=x-((a$="7") AND x>0)+
((a$="6") AND x<23)
4030 LET y=y-((a$="5") AND y>0)+
((a$="8") AND y<31)
4040 IF x+lx<22 THEN  GO SUB 500
0
4041 IF x+lx>21 THEN  GO SUB 500
1
4050 RETURN
4205 LET lx=lx-((a$="7") AND x+l
x>x)+((a$="6") AND x+lx<23)
4210 LET ly=ly-((a$="5") AND y+l
y>y)+((a$="8") AND y+ly<31)
4230 IF x+lx<22 THEN  GO SUB 500
0
4231 IF x+lx>21 THEN  GO SUB 500
1
4240 RETURN
5000 PRINT AT x+lx,y+ly; FLASH 8
; PAPER 8; INK 8; BRIGHT 8; OVER
 1; INVERSE 1;" ";: RETURN
5001 PRINT #0;AT x+lx-22,y+ly; F
LASH 8; PAPER 8; INK 8; BRIGHT 8
; OVER 1; INVERSE 1;" ";: RETURN

7999 STOP
8000 REM Stampa righe centrate
8010 LET py=(32-LEN a$)/2
8020 PRINT AT px,py;a$: GO SUB 8
030: RETURN
8030 PAPER 0: INK 7: INVERSE 0:
OVER 0: FLASH 0: BRIGHT 0: RETUR
N
9000 CLEAR 42999: LOAD *"m";1;"w
in$"CODE
9001 DATA 33,248,167,17,0,64,1,0
,27,237,176,201
9002 DATA 17,248,167,33,0,64,1,0
,27,237,176,201
9003 RESTORE 9001: FOR k=0 TO 23
: READ a: POKE 49950+k,a: NEXT k
9004 RUN
9010 REM L/M mem=>vid
9020 RANDOMIZE USR 49950
9030 RETURN
9040 REM L/M vid=>mem
9050 RANDOMIZE USR 49962
9060 RETURN
9900 SAVE *"m";1;"win$_gest" LIN
E 9000: VERIFY *"m";1;"win$_gest
"
9910 SAVE *"m";1;"win$"CODE 6450
0,1000: VERIFY *"m";1;"win$"CODE
 : STOP
9920 CLEAR : SAVE "win$_gest" LI
NE 9950: SAVE "win$"CODE 64500,1
000
9925 VERIFY "": VERIFY ""CODE
9930 STOP
9950 CLEAR 42999: LOAD ""CODE :
GO TO 9001
```

CALCOLO BALISTICO

```
  1 REM    CALCOLO BALISTICO

           di Perrella G.Luigi

  7 CIRCLE  INK 2;40,100,30: CI
RCLE  INK 4; OVER 0;80,70,30: CI
RCLE  INK 1; OVER 1;120,70,30: C
IRCLE  INK 5; OVER 0;160,70,30:
CIRCLE  INK 3; OVER 1;200,100,30
: PRINT AT 5,8;"B A L I C A L C"
: PAUSE 100
  9 BEEP .1,40: BEEP .1,20: BEE
P .1,40: PAUSE 10: BORDER 1: PAP
ER 1: INK 7: CLS
  10 PRINT  INK 6;"C A L C O L O
  B A L I S T I C O"
  11 PRINT  INK 5;"_____________
_____________"
  15 PRINT AT 20,5;"USIAMO IL SI
STEMA MKSA"
  20 PRINT AT 4,3;"INSERIRE I DA
TI CONOSCIUTI": BEEP .5,30
  21 PRINT
  30 PRINT "   VI  = ","M/S"
  31 INPUT A$: PRINT AT 6,9;A$
  32 PRINT "   U   = ","RAD"
  33 INPUT "PI/";U$: PRINT AT 7,
9;"PI/";U$
  34 PRINT "   T   = ","SEC"
  35 INPUT T$: PRINT AT 8,9;T$
  36 PRINT "   X(T)= ","M"
  37 INPUT B$: PRINT AT 9,9;B$
  38 PRINT "   Y(T)= ","M"
  39 INPUT C$: PRINT AT 10,9;C$
  40 PRINT "   VX  = ","M/S"
  41 INPUT D$: PRINT AT 11,9;D$
  42 PRINT "   VY  = ","M/S"
  43 INPUT E$: PRINT AT 12,9;E$
  46 PRINT "   YM  = ","M"
  47 INPUT G$: PRINT AT 13,9;G$
  48 PRINT "   TM  = ","SEC"
  49 INPUT H$: PRINT AT 14,9;H$
  50 PRINT "   TV  = ","SEC"
  51 INPUT I$: PRINT AT 15,9;I$
  52 PRINT "   XM  = ","M"
  53 INPUT L$: PRINT AT 16,9;L$
  54 BEEP .5,30: PRINT #1;"DATI
CONGRUENTI? (S/N)": PAUSE 0: IF
INKEY$="n" THEN  BEEP .5,10: GO
TO 9
  55 BORDER 1: PAPER 1: INK 7: C
LS : BEEP .1,40: BEEP .1,20: BEE
P .1,40
  60 PRINT "                MENU "
  61 PRINT
  62 PRINT "0 CALCOLO DELLA TRAI
ETTORIA"
  63 PRINT "1 CALCOLO DI X(T) E
Y(T),DATO T"
  64 PRINT "2 CALCOLO DELLA VELO
```

```
CITA' IN. VI"
  65 PRINT "3 CALCOLO DELL'ANGOL
O U"
  66 PRINT "4 CALCOLO DI T DATI
X(T) E Y(T)"
  67 PRINT "5 CALCOLO DI VX,VY,V
,DATO T"
  68 PRINT "6 CALCOLO DELLA QUOT
A MAX YM"
  69 PRINT "7 CALCOLO DEL TEMPO
REL. A YM"
  70 PRINT "8 CALCOLO DEL TEMPO
DI VOLO TV"
  71 PRINT "9 CALCOLO DELLA GITT
ATA MAX XM"
  72 PRINT
  73 PRINT "C CAMBIO PARAMETRI"
  74 PRINT "F FINE LAVORO"
  75 LET PA=500
  76 LET G=9.81
  77 INPUT S$
  78 IF S$="c" THEN  GO TO 9
  79 IF S$="f" THEN  GO TO 9200
  96 IF S$<"0" OR S$>"9" THEN  G
O TO 75
  97 LET S=VAL S$
  98 CLS
  99 LET P=(1000+(100*S))
 100 GO TO P
1001 REM EQ. TRAIETTORIA
1005 IF A$="" OR U$="" THEN  GO
TO 6000
1006 IF U$="0" THEN  GO TO 6500
1010 LET VI=VAL A$
1015 LET U=PI/VAL U$
1016 IF VI<=0 OR U=PI/2 THEN  PR
INT #1;"TRAIETTORIA NON CALCOLAB
ILE": PAUSE PA: GO TO 55
1017 PRINT AT 10,0;"Y= X";TAN U;
"-X^2*";G/(2*VI*VI*COS U*COS U)
1019 PAUSE 500
1020 PRINT "VUOI IL GRAFICO DELL
A TRAIETTO- RIA ?        (S/N)"
1021 INPUT Z$
1022 IF Z$<>"s" AND Z$<>"n" THEN
  GO TO 1021
1025 IF Z$="n" THEN  GO TO 55
1026 IF Z$="s" THEN  GO TO 9000
1100 REM X(T) E Y(T) DATO T
1101 IF B$<>"" THEN  GO TO 1180
1102 IF A$="" OR U$="" OR T$=""
THEN  GO TO 6000
1103 IF U$="0" THEN  GO TO 6500
1104 LET VI=VAL A$
1105 LET U=PI/VAL U$
1106 LET T=VAL T$
1107 LET XT=T*VI*COS U
1108 LET B$=STR$ XT
1109 CLS
1110 PRINT AT 8,5;"X(T)= ";XT;TA
B 25;"M"
1111 PRINT
1112 IF C$<>"" THEN  GO TO 1185
1113 IF A$="" OR U$="" OR T$=""
THEN  GO TO 6000: IF U$="0" THEN
  GO TO 6500
1115 LET U=PI/VAL U$
1120 LET T=VAL T$
1121 LET VI=VAL A$
1123 LET YT=(T*VI*SIN U)-((T*T*G
)/2)
1124 LET C$=STR$ YT
1125 PRINT AT 10,5;"Y(T)= ";YT;T
AB 25;"M"
1130 GO TO 7500
1180 PRINT AT 8,5;"X(T)= ";B$;TA
B 25;"M"
1181 PRINT
1182 GO TO 1112
1185 PRINT AT 10,5;"Y(T)= ";C$;T
AB 25;"M"
1186 GO TO 7500
1201 REM CALCOLO DI VI
1204 IF A$<>"" THEN  GO TO 1297
1205 IF U$="0" THEN  GO TO 6500
1206 PRINT "DA QUALI DATI PARTI?
"
1207 PRINT "A XM,U;B TV,U;C TM,U
;          D YM,U;E VX,U;F VY,U
,T"
1210 INPUT S$
1212 IF S$="a" THEN  GO TO 1230
1213 IF S$="b" THEN  GO TO 1240
1214 IF S$="c" THEN  GO TO 1250
1215 IF S$="d" THEN  GO TO 1260
1216 IF S$="e" THEN  GO TO 1270
1217 IF S$="f" THEN  GO TO 1280
1220 GO TO 1210
1230 IF L$="" OR U$="" THEN  GO
SUB 6000
1231 LET XM=VAL L$
1232 LET U=PI/VAL U$
1235 LET K=SQR ((XM*G)/SIN (2*U)
)
1236 GO TO 7000
1240 IF I$="" OR U$="" THEN  GO
TO 6000
1241 LET TV=VAL I$
1242 LET U=PI/VAL U$
1245 LET K=(TV*G)/(2*SIN U)
1246 GO TO 7000
1250 IF H$="" OR U$="" THEN  GO
TO 6000
1251 LET TM=VAL H$
1252 LET U=PI/VAL U$
1255 LET K=(TM*G)/SIN U
1256 GO TO 7000
1260 IF G$="" OR U$="" THEN  GO
TO 6000
1261 LET YM=VAL G$
1262 LET U=PI/VAL U$
1265 LET K=SQR ((2*YM*G)/(SIN U*
SIN U))
1266 GO TO 7000
1270 IF D$="" OR U$="" THEN  GO
TO 6000
1271 LET VX=VAL D$
1272 LET U=PI/VAL U$
1275 LET K=VX/COS U
1276 GO TO 7000
1280 IF E$="" OR U$="" OR T$=""
THEN  GO TO 6000
1281 LET VY=VAL E$
1282 LET T=VAL T$
1283 LET U=PI/VAL U$
1285 LET K=(VY+G*T)/SIN U
1286 GO TO 7000
1297 PRINT AT 10,5;"VI= ";A$;TAB
 25;"M/S"
1298 GO TO 7500
1301 REM CALCOLO DI U
1302 IF U$<>"" THEN  GO TO 1387
1303 PRINT "DA QUALI DATI PARTI?
"
1304 PRINT "A XM,VI;B TV,VI;C TM
,VI;          D YM,VI;E VX,VI;F VY
```

```
,VI,T"
1305 INPUT S$
1306 IF S$="a" THEN  GO TO 1330
1307 IF S$="b" THEN  GO TO 1340
1308 IF S$="c" THEN  GO TO 1350
1309 IF S$="d" THEN  GO TO 1360
1310 IF S$="e" THEN  GO TO 1370
1311 IF S$="f" THEN  GO TO 1380
1312 GO TO 1310
1330 IF L$="" OR A$="" THEN  GO
TO 6000
1331 LET XM=VAL L$
1332 LET VI=VAL A$
1335 LET H=0.5*ASN ((XM*G)/(VI*V
I))
1336 GO TO 7200
1340 IF I$="" OR A$="" THEN  GO
TO 6000
1341 LET TV=VAL I$
1342 LET VI=VAL A$
1345 LET H=ASN ((TV*G)/(2*VI))
1346 GO TO 7200
1350 IF H$="" OR A$="" THEN  GO
TO 6000
1351 LET TM=VAL H$
1352 LET VI=VAL A$
1355 LET H=ASN ((TM*G)/VI)
1356 GO TO 7200
1360 IF G$="" OR A$="" THEN  GO
TO 6000
1361 LET YM=VAL G$
1362 LET VI=VAL A$
1365 LET H=ASN SQR ((2*YM*G)/(VI
*VI))
1366 GO TO 7200
1370 IF D$="" OR A$="" THEN  GO
TO 6000
1371 LET VX=VAL D$
1372 LET VI=VAL A$
1375 LET H=ACS (VX/VI)
1376 GO TO 7200
1380 IF E$="" OR A$="" OR T$=""
THEN  GO TO 6000
1381 LET VY=VAL E$
1382 LET VI=VAL A$
1383 LET T=VAL T$
1385 LET H=ASN ((VY+G*T)/VI)
1386 GO TO 7200
1387 PRINT AT 10,5;"U = PI/";U$;
TAB 25;"RAD"
1388 GO TO 7500
1401 REM CALCOLO DI T PER X(T),
Y(T) DATI
1402 IF T$="" THEN  GO TO 1407
1403 CLS
1405 PRINT AT 10,5;"T= ";T$;TAB
25;"SEC"
1406 GO TO 7500
1407 IF U$="0" THEN  GO TO 6500
1450 PRINT "CONOSCI: A X(T);B Y(
T)"
1451 INPUT S$
1452 IF S$<>"a" AND S$<>"b" THEN
  GO TO 1451
1453 IF S$="b" THEN  GO TO 1469
1454 IF B$="" OR A$="" OR U$=""
THEN  GO TO 6000
1456 LET U=PI/VAL U$
1457 LET XA=VAL B$
1458 LET VI=VAL A$
1459 LET Q=XA/(VI*COS U)
1460 LET T$=STR$ Q
1467 PRINT AT 10,5;"T= ";Q;TAB 2
5;"SEC"
1468 GO TO 7500
1469 CLS
1470 IF C$="" OR U$="" OR A$=""
THEN  GO TO 6000
1471 LET YB=VAL C$
1475 LET U=PI/VAL U$
1478 LET VI=VAL A$
1480 LET Q=(VI*SIN U-SQR (VI*VI*
SIN U*SIN U-2*G*YB))/G
1485 PRINT AT 10,5;"T= ";Q;TAB 2
5;"SEC"
1486 LET T$=STR$ Q
1490 GO TO 7500
1501 REM CALCOLO DI VX,VY,V DATO
 T
1502 CLS
1503 IF D$<>"" AND E$<>"" THEN
GO TO 1582
1505 IF A$="" OR U$="" OR T$=""
THEN  GO TO 6000
1506 IF U$="0" THEN  GO TO 6500
1520 LET VI=VAL A$
1521 LET U=PI/VAL U$
1522 LET T=VAL T$
1530 LET M=VI*COS U
1531 LET D$=STR$ M
1535 LET N=VI*SIN U-(G*T)
1536 LET E$=STR$ N
1540 LET VE=SQR (M*M+N*N)
1550 PRINT AT 8,5;"VX= ";M;TAB 2
5;"M/S"
1553 PRINT
1555 PRINT AT 10,5;"VY= ";N;TAB
25;"M/S"
1557 PRINT
1560 PRINT AT 12,5;"V = ";VE;TAB
 25;"M/S"
1580 GO TO 7500
1582 LET M=VAL D$
1583 LET N=VAL E$
1584 LET VE=SQR (M*M+N*N)
1585 CLS
1586 PRINT AT 8,5;"VX= ";D$;TAB
25;"M/S"
1587 PRINT
1588 PRINT AT 10,5;"VY= ";E$;TAB
 25;"M/S"
1589 PRINT
1590 PRINT AT 12,5;"V= ";VE;TAB
25;"M/S"
1591 GO TO 7500
1601 CLS
1602 REM CALCOLO DI YM QUOTA MAX
1610 IF G$="" THEN  GO TO 1621
1615 PRINT AT 10,5;"YM= ";G$;TAB
 25;"M"
1620 GO TO 7500
1621 IF U$="0" THEN  GO TO 6500
1649 IF A$="" OR U$="" THEN  GO
TO 6000
1650 LET VI=VAL A$
1651 LET U=PI/VAL U$
1655 LET J=(VI*VI*SIN U*SIN U)/(
2*G)
1656 LET G$=STR$ J
1660 PRINT AT 10,5;"YM= ";J;TAB
25;"M"
1665 GO TO 7500
1701 CLS
1702 REM CALCOLO DI TM
1706 IF H$="" THEN  GO TO 1748
1708 PRINT AT 10,5;"TM= ";H$;TAB
```

```
 25;"SEC"
1710 GO TO 7500
1748 IF A$="" OR U$="" THEN  GO
TO 6000
1749 IF U$="0" THEN  GO TO 6500
1750 LET VI=VAL A$
1751 LET U=PI/VAL U$
1755 LET W=(VI*SIN U)/G
1756 LET H$=STR$ W
1760 PRINT AT 10,5;"TM= ";W;TAB
25;"SEC"
1770 GO TO 7500
1801 CLS
1802 REM CALCOLO DI TV
1806 IF I$="" THEN  GO TO 1848
1808 PRINT AT 10,5;"TV= ";I$;TAB
 25;"SEC"
1810 GO TO 7500
1848 IF A$="" OR U$="" THEN  GO
TO 6000
1849 IF U$="0" THEN  GO TO 6500
1850 LET VI=VAL A$
1851 LET U=PI/VAL U$
1855 LET WR=2*(VI*SIN U)/G
1856 LET I$=STR$ WR
1858 PRINT AT 10,5;"TV= ";WR;TAB
 25;"SEC"
1860 GO TO 7500
1901 CLS
1902 REM CALCOLO DI XM
1910 IF L$="" THEN  GO TO 1948
1915 PRINT AT 10,5;"XM= ";L$;TAB
 25;"M"
1920 GO TO 7500
1948 IF A$="" OR U$="" THEN  GO
TO 6000
1949 IF U$="0" THEN  GO TO 6500
1950 LET VI=VAL A$
1951 LET U=PI/VAL U$
1955 LET GI=(VI*VI*SIN (2*U))/G
1956 LET L$=STR$ GI
1960 PRINT AT 10,5;"XM= ";GI;TAB
 25;"M"
1965 GO TO 7500
6000 REM ERRORE
6005 PRINT  FLASH 1;"  ERRORE ER
RORE ERRORE ERRORE "
6006 PRINT
6007 PRINT
6008 PRINT
6019 PRINT TAB 5;"VI     = ";A$
6020 IF U$="" THEN  PRINT TAB 5;
"U      = ": GO TO 6022
6021 PRINT TAB 5;"U      = ";"PI/
";U$
6022 PRINT TAB 5;"T      = ";T$
6023 PRINT TAB 5;"X(T)   = ";B$
6024 PRINT TAB 5;"Y(T)   = ";C$
6025 PRINT TAB 5;"VX     = ";D$
6026 PRINT TAB 5;"VY     = ";E$
6028 PRINT TAB 5;"YM     = ";G$
6029 PRINT TAB 5;"TM     = ";H$
6030 PRINT TAB 5;"TV     = ";I$
6031 PRINT TAB 5;"XM     = ";L$
6032 FOR p=1 TO 5: BEEP .4,10: B
EEP .4,20: NEXT p
6040 PRINT #1;"I DATI SONO INSUF
FICIENTI,TORNO ALLA FASE DI IMMI
SSIONE DATI"
6045 PAUSE PA
6050 GO TO 9
6500 PRINT #1;"ANGOLO INDEFINITO
": PAUSE PA: GO TO 6000
```

```
7000 REM STAMPA VI
7001 CLS
7005 PRINT AT 10,5;"VI= ";K;TAB
25;"M/S"
7006 LET A$=STR$ K
7008 GO TO 7500
7200 REM STAMPA U
7201 CLS
7205 PRINT AT 10,5;"U= ";H,"PI/"
;PI/H;TAB 25;"RAD"
7206 LET U$=STR$ H
7210 GO TO 7500
7500 REM PAUSA LETTURA
7505 PAUSE 0
7508 GO TO 55

9000 REM DISEGNO TRAIETTORIA
9005 IF VI<0 THEN  PRINT #1;"VEL
OCITA' NEGATIVA !": GO TO 7500
9006 IF U<0 OR U>PI/2 THEN  PRIN
T #1;"ANGOLO NEGATIVO !": GO TO
7500

9007 CLS
9020 PRINT "QUESTA E' LA TRAIETT
ORIA:"
9021 PRINT "PER VI= ";A$;TAB 22;
"M/S"
9022 PRINT "PER U = PI/";U$;TAB
22;"RAD"
9023 LET VI=VAL A$
9024 LET U=PI/VAL U$
9030 FOR I=0 TO 120
9035 PLOT  BRIGHT 1;0,I
9040 NEXT I
9045 FOR I=0 TO 240
9050 PLOT  BRIGHT 1;I,0
9055 NEXT I
9056 PRINT AT 5,0;"Y";AT 21,31;"
X"
9060 FOR X=0 TO 240
9065 LET Y=(X*TAN U)-(X*X*G)/(2*
VI*VI*COS U*COS U)

9066 IF Y>150 THEN  PRINT #1;"TR
AIETTORIA FUORI SCALA": COPY : G
O TO 7500
9067 IF Y<0 AND X=0 THEN  PRINT
#1;"ANGOLO NEGATIVO !": GO TO 75
00
9068 IF Y<=0 AND X>0 THEN  BEEP
1,40: COPY : GO TO 7500
9070 PLOT  BRIGHT 1;X,Y

9100 NEXT X
9105 BEEP 1,40
9106 COPY
9110 GO TO 7500
9200 CLS : BEEP .1,40: BEEP .1,2

0: BEEP .1,40: PRINT AT 10,2; FL
ASH 1;"E VA BENE,TOLGO IL DISTUR
BO!": BRIGHT 1: PRINT AT 0,14;"_
_______": PLOT 152,104: DRAW 24,
24: DRAW 0,40: DRAW -24,-24: PLO
T 88,144: DRAW 24,24

9210 FOR p=50 TO 0 STEP -1: PRIN
T  INK RND*8; INVERSE 1;AT 4,11;
"          ";AT 8,11;"          ";AT
 5,11;" ";AT 6,11;" ";AT 7,11;"
";AT 5,18;" ";AT 6,18;" ";AT 7,1
8;" ": PRINT  INVERSE 1; BRIGHT
1;AT 6,14;p: BEEP 0.05,P: NEXT p
: NEW
```

IL PROBLEMA DELLE PROVE RIPETUTE

```
  1 REM
  2 REM  PROVE RIPETUTE II
  3 REM  3-04-1987   V 2.0
  4 REM  by BIANCHI/PADOVANI
  5 REM
  7 DIM y(34): DIM g(34)
  8 REM  MENU'
 10 LET Y=1: LET N=20: LET P=.5

 15 BORDER 0: BRIGHT 1: PAPER 1
: INK 6: CLS : PRINT AT 2,1;"PRO
BLEMA DELLE PROVE RIPETUTE"
 20 PRINT AT 5,3;"1--->K SUCCES
SI"''"   2--->ALMENO K SUCCESSI"
''"   3--->AL MASSIMO K SUCCESSI
"''"   4--->DA K A K1 SUCCESSI"'
'"   5--->CAMBIA P e N"
 22 PRINT '"   6--->DISEGNA GRA
FICO"
 25 LET Q=1-P: PRINT AT 18,5;"N
UMERO DELLE PROVE:";N'';"  P=";P
;TAB 19;"Q=";Q;#1;" Program by M
ARIO BIANCHI ?1986"
 30 LET YR=154-16*Y
 40 PLOT 22,YR: DRAW 211,0: DRA
W 0,-13: DRAW -211,0: DRAW 0,12

 50 LET A$=INKEY$: IF A$="" THE
N  GO TO 50
 60 IF A$<>CHR$ 13 AND A$<>"6"
AND A$<>"7" AND A$<>"0" THEN  GO
 TO 50
 65 BEEP .0125,20
 70 LET Y=Y+(A$="6" AND Y<6)-(A
$="7" AND Y>1)

 80 IF A$="0" OR A$=CHR$ 13 THE
N  GO TO Y*1000-(900 AND y=6)
 90 INVERSE 1: PLOT 22,YR: DRAW
 211,0: DRAW 0,-13: DRAW -211,0:
 DRAW 0,12: INVERSE 0
 100 GO TO 30
 999 REM  K SUCCESSI

1000 GO SUB 5020: LET I=K: GO SU
B 8000  LET A=G: GO TO 9000
1999 REM  ALMENO K SUCCESSI

2000 GO SUB 5020: LET A=0: FOR I
=K TO N: GO SUB 8000: LET A=A+G:
 NEXT I: GO TO 9000
2999 REM  FINO A K SUCCESSI
3000 GO SUB 5020: LET A=0: FOR I
=0 TO K: GO SUB 8000: LET A=A+G:
 NEXT I: GO TO 9000
3999 REM  DA K A K1 SUCCESSI
4000 GO SUB 5020: GO SUB 5040: L
ET A=0: FOR I=K TO K1: GO SUB 80
00: LET A=A+G: NEXT I: GO TO 900
0
4999 REM  CAMBIA P e N
5000 INPUT "P=";P: IF P>1 OR P<0
 THEN  GO TO 5000
5010 INPUT "N=";N: IF N<>INT N O
R N<1 OR N>30 THEN  GO TO 5010
5015 GO TO 15
5019 REM  CAMBIA K
5020 INPUT "K=";K: IF K<>INT K O
R K>N OR K<0 THEN  GO TO 5020


5030 RETURN
5039 REM  CAMBIA K1
5040 INPUT "K1=";K1: IF K1<>INT
K1 OR K1<=K OR K1>N THEN  GO TO

5040
5050 RETURN
5099 REM  DISEGNA GRAFICO

5100 LET max=-1: LET x=8
5110 GO SUB 5020: GO SUB 5040
5120 LET sx=INT (248/(k1-k+1))
5130 CLS : PRINT '''"     ATTENDI
,STO CALCOLANDO:"

5140 FOR i=k TO k1: PRINT AT 3,2
7;i: GO SUB 8000: LET g(i-k+1)=g
: IF g(i-k+1)>max THEN  LET max=
g(i-k+1)


5150 NEXT i: GO SUB 5700
5160 FOR f=1 TO (k1-k+1): LET y(
f)=g(f)*158/max

5170 GO SUB 5800: NEXT f
5199 REM  UNISCE I PUNTI
5205 LET y1=0: PLOT 9,9: LET x=I
NT (sx/2+.5): FOR f=x TO 247 STE
P sx
5210 DRAW  OVER 1;x,(y((f-x)/sx+
1)-y1): LET y1=y((f-x)/sx+1): LE
T x=sx: NEXT f
5215 DRAW  OVER 1;255-PEEK 23677
,9-PEEK 23678

5220 GO TO 9010
5699 REM  PREDISPONE GLI ASSI
5700 CLS : PLOT 7,0: DRAW 0,167:
 PLOT 0,7: DRAW 255,0
5705 FOR f=8+sx TO 255 STEP sx:
PLOT f,5: PLOT f,6: NEXT f
5710 FOR f=23 TO 167 STEP 16: PL
OT 6,f: PLOT 5,f: NEXT f
5720 PRINT AT 0,3;"VALORE MASSIM
O:";max
5780 RETURN

5799 REM  COLORA I RETTANGOLI
5800 FOR i=x+1 TO x+sx-1: PLOT i
,9: DRAW 0,y(f): NEXT i: LET x=x
+sx: RETURN
6000 LET F1=1: IF F=0 THEN  RETU
RN
6010 FOR Z=2 TO F: LET F1=F1*Z:
NEXT Z: RETURN

7000 LET F=N: GO SUB 6000: LET B
=F1: LET F=I: GO SUB 6000: LET C
=F1: LET F=N-I: GO SUB 6000: LET
 C=B/(C*F1): RETURN
8000 GO SUB 7000: LET G=P^I*Q^(N
-I)*C: RETURN
9000 CLS : PRINT AT 3,2;"PROBABI
LITA'=";A
9010 PRINT #1;"  PREMI UN TASTO
PER IL MENU'": PAUSE 0: GO TO 15
9999 SAVE "PROVE RIP " LINE 1
```

IL MENU' DELL'ATARI ST

```
' Rem
'
' Programma  d'esempio
' scritto in Gfa-Basic
' per  la gestione dei
' menu' sull'Atari St.
'
@Init
Do
  On Menu
Loop
End
'
' Inizializzazione delle
' variabili
'
Procedure Init
  Cls
  Vie=7
  Dim Scelta$(Vie)
  Restore Dati_menu
  For X=Ø To Vie
    Read Scelta$(X)
  Next X
  Defmouse 3
  Menu Scelta$()
  On Menu  Gosub Main
Return
'
' Routine principale per
' la gestione del menu'
'
Procedure Main
  Scd$=Scelta$(Menu(Ø))
  If Scd$="INFO"
    @Desk
  Endif
  If Scd$="LOAD"
    @Load
  Endif
  If Scd$="SAVE"
    @Save
  Endif
  If Scd$="QUIT"
    @Quit
  Endif
  Defmouse 3
  Cls
  Menu Scelta$()
Return
'
' Procedura associata
' all'opzione "Desk"
'
Procedure Desk
  St$="SYSTEMS  EDITORIALE | "
  Alert Ø,St$,1," OK ",X
Return
'
' Procedura associata
' all'opzione "Load"
'
Procedure Load
  Print At(33,3);"Procedura LOAD"
  Fileselect "A:\*.*","",F$
Return
'
' Procedura associata
' all'opzione "Save"
'
Procedure Save
  Print At(33,3);"Procedura SAVE"
  Fileselect "A:\*.*","",F$
Return
'
' Procedura associata
' all'opzione "Quit"
'
Procedure Quit
  St$="  SEI SICURO   "
  Alert 2,St$,1," YES | NO  ",X
  If X=1
    End
  Endif
Return
'
Dati_menu:
Data "DESK","INFO",""
Data "FILE","LOAD","SAVE","QUIT",""
'
```

ZETAGRAF

```
  *** ZETAGRAF by Zubani Roberto e
Lorenzo/ SECONDA PARTE ***
  ** riserva una porzione di memoria,
```

```
che viene visualizzata su schermo, in
cui scrivere eventuali appunti **
Procedure Note
  Openw 0
  @Box(192,255)
  Deftext 1,0,0,6
  For T=9 To 39
    Print At(31,T);No$(T-9)
  Next T
  For T=9 To 39
    Print At(31,T);
    Form Input 20 As No$(T-9)
  Next T
  Closew 0
  Sput Video$
  @Standard
Return
' ** consente il disegno speculare
secondo varie direzioni **
Procedure Spec
  @Utente
  Alert 2,"Orizzontale, verticale|o
entrambi ?",1,"Oriz.|Vert.|Entr.",S%
  Repeat
    Mouse X%,Y%,K%
    If K%=1
      Plot X%,Y%
      If S%=1 Or S%=3
        Plot 640-X%,Y%
      Endif
      If S%=2 Or S%=3
        Plot X%,381-Y%
      Endif
      If S%=3
        Plot 640-X%,381-Y%
      Endif
    Endif
  Until K%=2
  @Standard
Return
' ** genera archi di circonferenza di
ampiezza fissata a piacere **
Procedure Arc
  Sget Video$
  @Box(180,80)
  @Input_at(31,12,3,"Gradi:  da ","",
*Grai)
  @Input_at(40,13,3,"a ","",*Graf)
  Sput Video$
  @Utente
  Defmouse 7
  @Mouse
  Cx=X
  Cy=Y
  Out 2,7
  Pause 10
  @Mouse
Do
  Exit If Mousek=2
  Sput Video$
    Ellipse Cx,Cy,Abs(Cx-Mousex),
Abs(Cy-Mousey),Grai*10,Graf*10
  Loop
  If Ty=2
    Pellipse Cx,Cy,Abs(Cx-Mousex),
Abs(Cy-Mousey),Grai*10,Graf*10
  Endif
  @Standard
Return

  ** genera poligoni regolari,
pentagono, esagono e ottagono per il
momento **

Procedure Pol
  Alert 0,"Quale tipo di poligono
vuoi?",0,"PENTA|ESAGONO|OTTAGONO",T
  N1=4+T-(T=3)
  @Mouse
  X1=X
  Y1=Y
  Do
    @Mouse
    Exit If K=2
    R=Sqr((X1-X)^2+(Y1-Y)^2)
    For I=0 To N1
      X(I)=X1+R*Sin(I*2*Pi/N1)
      Y(I)=Y1+R*Cos(I*2*Pi/N1)
    Next I
    Sput Video$
    Polyline N1+1,X(),Y()
  Loop
  Sput Video$
  @Utente
  @Ty_pol
  @Standard
Return

' ** consente di definire gli estremi
delle linee, a freccia, arrotondati
ecc **

Procedure Estr
  For In_1=0 To 2
    @Com3("Inizio=",In_1)
    Exit If K=2
  Next In_1
  In_1=In_1*-(K=2)
  Pause 4
  For Fi_1=0 To 2
    @Com3("Fine=",Fi_1)
    Exit If K=2
  Next Fi_1
  Sput Video$
  @Standard
Return
' ** definizione a piacere di linea **
Procedure Defl
  Sget Video$
  Do
    @Box(180,80)
```

```
    @Us_line(Use_st)
    Defline 7
    Line 248,210,388,210
    Print At(32,11);"Linea utente(0-
1):"
    @Input_at(32,12,18,"",
"&X"+Bin$(Use_st),*K)
    Exit If K=Use_st
    Use_st=K
  Loop
  Sput Video$
  @Standard
Return
' ** definizione a piacere di pattern
**
Procedure Defp
  @Com4
  T=1
  For Ri=R+1 To R+16
    P$(T)="&x"
    For Co=C+1 To C+16
      P$(T)=P$(T)+Str$(Point(Co,Ri))
    Next Co
    Inc T
  Next Ri
  Pd$=""
  For T=1 To 16
    Pd$=Pd$+Mki$(Val(P$(T)))
  Next T
  Deffill Col,Pd$
  St_p=4
  Pat=0
Return
' ** gestisce la possibilita' di
eliminare i bordi delle figure
geometriche **
Procedure Pen
  @Utente
  @Edge(0)
  Repeat
    Mouse X%,Y%,K%
    If K%=1
      Pbox X%,Y%,X%+8,Y%+8
    Endif
  Until K%=2
  @Standard
Return
' ** muove su video un cursore a
croce riportando in tempo reale le
sue coordinate **
Procedure Mis
  Alert 2,"Tolgo i pixel!del menu ?",
1," NO ! SI ",P
  P=18*(P-1)
  Hidem
  Graphmode 3
  Mouse X,Y,K
  Get 10,370,120,399,T$
  Line X,0,X,399
  Line 0,Y,640,Y
  Repeat
    If X<>Mousex Or Y<>Mousey
      Print At(5,25);Using "x=###
y=###",Mousex,Mousey-P;
      Line X,0,X,399
      Line 0,Y,640,Y
      X=Mousex
      Y=Mousey
      Line X,0,X,399
      Line 0,Y,640,Y
    Endif
    If Mousek=1
      Plot Mousex,Mousey
    Endif
  Until Mousek=2
  Line X,0,X,399
  Line 0,Y,640,Y
  Showm
  Put 10,370,T$,3
  @Standard
Return
' ** visualizza su schermo i codici
ascii deL SET DI CARATTERI
disponibile **
Procedure Ascii
  Do
    A$=""
    B$=""
    For C=Car To Car+3
      A$=A$+Chr$(C)+"="+Str$(C)+"   "
      B$=B$+Chr$(C+4)+"="+Str$(C+4)+"
 "
    Next C
    Alert 0,A$+"!"+String$(27,"-
")+"!"+B$+"!",3,"   !   !STOP",X
    Exit If X=3
    If X=1
      Sub Car,8
      If Car<0
        Car=248
      Endif
    Endif
    If X=2
      Add Car,8
      If Car>248
        Car=0
      Endif
    Endif
  Loop
Return
' ** gestisce una lente di
ingrandimento **
Procedure Lent
  Do
    Color 1
    Hidem
    Graphmode 3
    Mouse C,R,K
    Box C,R,C+68,R+49
    Repeat
```

```
      If C<>Mousex Or R<>Mousey
        Box C,R,C+68,R+49
        Mouse C,R,K
        Box C,R,C+68,R+49
      Endif
    Until Mousek
    Exit If Mousek=2
    Box C,R,C+68,R+49
    Sget Err$
    Get C+1,R+1,C+67,R+48,Ar$
    Cls
    Put 10,10,Ar$,3
    Showm
    Deftext 1,0,0,4
    Graphmode 1
    For X%=96 To 632 Step 8
      Line X%,8,X%,392
    Next X%
    For Y%=8 To 393 Step 8
      Line 96,Y%,632,Y%
    Next Y%
    For X%=0 To 15
      Text 98+X%*8,6,0,
Right$(Str$(X%))
      Text 88,14+X%*8,0,
Right$(Str$(X%))
    Next X%
    Deffill 1,1
    For Y%=10 To 57
      For X%=10 To 76
        If Point(X%,Y%)
          Xh%=96+(X%-10)*8
          Yh%=8+(Y%-10)*8
          Pbox Xh%,Yh%,Xh%+8,Yh%+8
        Endif
      Next X%
    Next Y%
    Print At(2,10);">ESCI"
    Do
      @Mouse
      Deffill 1,2-K,2-K
      Exit If K And X<50
      If X<632 And Y>8 And Y<392 And
X>96
        Xh%=Fix(X/8)*8
        Yh%=Fix(Y/8)*8
        Pbox Xh%,Yh%,Xh%+8,Yh%+8
        Color 2-K
        Plot 10+(Xh%-96)/8,10+(Yh%-
8)/8
      Endif
    Loop
    Get 10,10,76,57,Ar$
    Sput Video$
    Put C+1,R+1,Ar$,3
    Sget Video$
    Pause 10
  Loop
  Sput Video$
  Showm
  @Standard
Return
' ** genera su schermo righelli in
scala centimetrica **
Procedure Righ
  Line 15,15,639,15
  Line 15,15,15,399
  Deftext 1,0,0,4
  For T=15 To 639 Step 3
    Plot T,14
    Plot 14,T
    If T Mod 30
      Line T,13,T,12
      Line 13,T,12,T
    Endif
  Next T
  For T=1 To 40
    Text T*30-3,10,0,Str$(T)
    Text 12-Len(Str$(T))*6,T*30+2,0,
Str$(T)
  Next T
Return
Scelta:
Data Desk,  Zetagraf V 4.0
Data --------------------
Data 1,2,3,4,5,6,""
Data File, Salva disegno , Carica
disegno , Elimina file, Cancella
video, Stampa, Fine,----------------,
Note, Utente,""
Data Modi, Gomma, Disegna, Specchio,
Spruzzo, Linee, Linee unite ,  *Tipo*,
 Rettangoli, Cerchi, Archi, Poligoni,
""
Data Formato, *Errore*, Modo graf.,
Colore,----Linee----, Spessore, Stile,
 Estremi
Data Def Linea,---Retini----, Tipo,
Riempi, Def Retino, Pennello,""
Data Testi, Altezza, Stile, Rotazione
, Scrivi,-----------, Misure, ASCII,
Righelli, Lente,""
Data Blocchi, -Definire, -Muovere, -
Ruotare 90°, -Inversione, -Dimensioni,
 -Distorsione, -Inv. Pixel
Data  -Memorizzare , -Prendere, -
Salvare, -Caricare,""
Data Sprite, -Def Sprite, -Def
Maschera , -Def Dati, -Memorizza, -
Muovi, -Stampa dati, -Salvare , -
Caricare
Data "","",*
Procedure File
  If Instr(F$,".")
    F$=Left$(F$,Instr(F$,".")-1)
  Endif
Return
Procedure Standard
  Defmouse 0
  Deffill 1,0,0
```

```
  Defline 1,1,0,0
  Deftext 1,0,0,13
  Graphmode 1
  @Edge(1)
  Color 1
Return
Procedure Utente
  Color Col
  Deffill Col,St_p,Pat
  Defline St_l,Sp,In_l,Fi_l
  Deftext Col,St_t,Rot,Alt
  @Edge(Ed)
  Graphmode Mg
Return
Procedure Mouse
  Repeat
    Mouse X,Y,K
  Until K
Return
Procedure Com2(A$,V)
  @Box(295,295)
  Deftext 1,St_t,Rot,Alt
  Text 320,190,0,A$+Str$(V)
  @Mouse
Return
Procedure Com3(A$,V)
  @Box(200,100)
  Defline St_l,Sp,In_l,Fi_l
  Line 230,190,410,190
  Text 230,220,0,A$+Str$(V)
  @Mouse
  @Standard
Return
Procedure Com4
  Hidem
  Graphmode 3
  Mouse C,R,K
  Box C,R,C+17,R+17
  Repeat
    If C<>Mousex Or R<>Mousey
      Box C,R,C+17,R+17
      C=Mousex
      R=Mousey
      Box C,R,C+17,R+17
    Endif
  Until Mousek=2
  Box C,R,C+17,R+17
  Showm
  @Standard
Return
Procedure Com5
  Sget V$
  S%=0
  Do
    X%=Mousex
    @Mouse
    Sput V$
    Exit If K=2
    S%=S%+Sgn(X%-Mousex)
    Draw 320+S%,10 To 320+S%+La%,10
To 320-S%+La%,10+Al% To 320-S%,10+Al%
To 320+S%,10
  Loop
Return
Procedure Com6
  Sget V$
  S%=0
  Do
    Y%=Mousey
    @Mouse
    Sput V$
    Exit If K=2
    S%=S%+Sgn(Y%-Mousey)
    Draw 320,10+S% To 320+La%,10-S%
To 320+La%,10+Al%-S% To 320,10+Al%+S%
To 320,10+S%
  Loop
Return
Procedure Sicuro
  Alert 2,"Sei sicuro?",1," SI : NO ",
Si
Return
Procedure Ty_ret
  If T=1
    If Ty=1
      Box X,Y,X1,Y1
    Else
      Pbox X,Y,X1,Y1
    Endif
  Else
    If Ty=1
      Rbox X,Y,X1,Y1
    Else
      Prbox X,Y,X1,Y1
    Endif
  Endif
Return
Procedure Ty_cer
  If Ty=2
    Pellipse X,Y,Abs(X-Mousex),Abs(Y-
Mousey)
  Else
    Ellipse X,Y,Abs(X-Mousex),Abs(Y-
Mousey)
  Endif
Return
Procedure Ty_pol
  If Ty=2
    Polyfill N1+1,X(),Y()
  Else
    Polyline N1+1,X(),Y()
  Endif
Return
Procedure Input_at(Col,Rig,Size,W$,Z$,
Pt)
  Repeat
    Print At(Col,Rig);W$;
    Form Input Size As Z$
  Until (Len(Z$) And Val?(Z$)=Len(Z$))
  *Pt=Val(Z$)
```

```
Return
Procedure Comis
  @Box(200,120)
Return
Procedure Box(L%,A%)
  Deffill 1,0,0
  X%=320-L%/2
  Y%=190-A%/2
  Defline 1,3
  Pbox X%-5,Y%-5,X%+L%+5,Y%+A%+5
  Box X%,Y%,X%+L%,Y%+A%
  @Standard
Return
Procedure Errore
  Alert 3,"Errore numero:"+Str$(Err),
1," OK !STOP",T
  If T=1
    On Error Gosub Errore
  Endif
  Stop
Return
Procedure Edge(V%)
  Dpoke Contrl,104
  Dpoke Contrl+2,0
  Dpoke Contrl+4,1
  Dpoke Contrl+6,0
  Dpoke Intin,V%
  Vdisys 104
Return

Procedure Us_line(P%)
  Dpoke Contrl,113
  Dpoke Contrl+2,0
  Dpoke Contrl+4,1
  Dpoke Contrl+6,0
  Dpoke Intin,P%
  Vdisys 113
Return
```

VIDEO A 160 COLONNE

```
' Rem
'
' ---------------------------
' Programma in Gfa-basic per
' Atari ST.
' Permette di visualizzare
' un file (formato Ascii) in
' pagina grafica, su una
' schermata di 160 colonne
' per 66 righe.
' ---------------------------
'
@Init
Repeat
  @Readchar
  @Ascii
  @Plotchar
Until Eof(#1)
'
X=140
Y=65
X$="  premi un tasto  "
@Hprint
@Getkey
'
End
'
' Getkey: attende che sia prem
uto un tasto '
'
Procedure Getkey
  Repeat
    I$=Inkey$
  Until I$<>""
Return
'
' Hprint: visualizza la string
a X$ alla posizione X, Y '
'
Procedure Hprint
  For I=1 To Len(X$)
    A$=Mid$(X$,I,1)
    @Ascii
    @Plotchar
  Next I
Return
'
' Init: inizializza le variabi
li e seleziona il file da elabo
rare '
Procedure Init
  Dim A(60,4,6)
  Cls
  Print
  Dir "*.*"
  Repeat
    Print
    Print "Nome file da legger
e"
    Print
    Form Input 12,A$
  Until Exist(A$)
  Open "i",#1,A$
  Cls
  Maxcol=160-1
  For C=1 To 59
    For Y=1 To 6
      For X=1 To 4
```

```
        Read D
        A(C,X,Y)=D
      Next X
    Next Y
  Next C
  X=0
  Y=0
Return
'
' Readchar: preleva un dato da
l file di input '
'
Procedure Readchar
  A$=Chr$(Inp(#1))
Return
'
' Ascii: converte A$ in caratt
ere stampabile Ascii '
'
Procedure Ascii
  If A$>="a" And A$<="z"
    A$=Chr$(Asc(A$)-32)
  Endif
  If A$>=" " And A$<="Z"
    Ch=Asc(A$)-32+1
  Else
    Ch=1
  Endif
Return
'
' Plotchar: disegna il caratte
re "CH" alla posizione X, Y '
'
Procedure Plotchar
  @Contret.
  If Flret=0 And A$<>Chr$(10)
    Xpos=X*4
    Ypos=Y*6
    For X1=1 To 4
      For Y1=1 To 6
        Color A(Ch,X1,Y1)
        Plot Xpos+X1,Y1+Ypos
      Next Y1
    Next X1
    Inc X
    If X Mod Maxcol=0
      X=0
      If Y<65
        Inc Y
      Endif
    Endif
  Endif
Return
'
' Contret: controlla se il cod
ice in ingresso e' un "new line
" '
'
Procedure Contret
  Flret=0
  If A$=Chr$(13)
    Flret=1
    X=0
    If Y<65
      Inc Y
    Endif
  Endif
Return
'
' Definizione di tutti i carat
teri satampabili '
'
' Carattere [ ]
'
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [!]
'
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,1,0,0
Data 0,0,0,0
'
' Carattere ["]
'
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [#]
'
Data 0,1,1,0
Data 1,1,1,1
Data 0,1,1,0
Data 1,1,1,1
Data 0,1,1,0
Data 0,0,0,0
'
```

```
' Carattere [$]
'
Data 0,1,1,0
Data 1,1,0,0
Data 0,1,1,0
Data 0,1,0,1
Data 1,1,1,0
Data 0,1,0,0
'
' Carattere [%]
'
Data 1,0,1,0
Data 0,0,1,0
Data 0,1,0,0
Data 1,0,0,0
Data 1,0,1,0
Data 0,0,0,0
'
' Carattere [&]
'
Data 0,1,0,0
Data 1,0,1,0
Data 0,1,0,0
Data 1,0,1,1
Data 0,1,1,0
Data 0,0,0,1
'
' Carattere [']
'
Data 0,0,1,0
Data 0,0,1,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [(]
'
Data 0,0,1,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,1,0
Data 0,0,0,0
'
' Carattere [)]
'
Data 0,1,0,0
Data 0,0,1,0
Data 0,0,1,0
Data 0,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
' Carattere [*]
'
Data 1,0,0,1
Data 0,1,1,0
Data 1,1,1,1
Data 0,1,1,0
Data 1,0,0,1
Data 0,0,0,0
'
' Carattere [+]
'
Data 0,0,0,0
Data 0,1,0,0
Data 1,1,1,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [']
'
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,1,0,0
Data 1,0,0,0
'
' Carattere [-]
'
Data 0,0,0,0
Data 0,0,0,0
Data 1,1,1,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [.]
'
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,0,0,0
Data 0,1,0,0
Data 0,0,0,0
'
' Carattere [/]
'
Data 0,0,0,1
Data 0,0,1,0
Data 0,1,1,0
Data 0,1,0,0
Data 1,0,0,0
Data 0,0,0,0
'
' Carattere [0]
'
Data 1,1,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [1]
'
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,0,0
'
' Carattere [2]
'
Data 1,1,1,0
Data 0,0,1,0
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [3]
'
Data 1,1,1,0
Data 0,0,1,0
Data 1,1,1,0
Data 0,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [4]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,1,0
Data 0,0,1,0
Data 0,0,0,0
'
' Carattere [5]
'
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,1,0
Data 0,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [6]
'
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [7]
'
Data 1,1,1,0
Data 0,0,1,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,0,0
'
' Carattere [8]
'
Data 1,1,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [9]
'
Data 1,1,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'
' Carattere [:]
'
Data 0,0,0,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [;]
'
Data 0,0,0,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,1,0,0
Data 1,0,0,0
Data 0,0,0,0
'
```

```
' Carattere [<]
'
Data 0,0,1,0
Data 0,1,0,0
Data 1,0,0,0
Data 0,1,0,0
Data 0.0.1,0
Data 0,0,0,0
'
' Carattere [=]
'
Data 0,0,0,0
Data 1,1,1,0
Data 0,0,0,0
Data 1,1,1,0
Data 0,0,0,0
Data 0,0,0,0
'
' Carattere [>]
'
Data 1,0,0,0
Data 0,1,0,0
Data 0,0,1,0
Data 0,1,0,0
Data 1,0,0,0
Data 0,0,0,0
'
' Carattere [?]
'
Data 0,1,0,0
Data 1,0,1,0
Data 0,0,1,0
Data 0,1,0,0
Data 0,0,0,0
Data 0,1,0,0
'
' Carattere [@]
'
Data 0,1,1,0
Data 1,0,0,1
Data 1,1,1,1
Data 1,1,1,0
Data 1,0,0,0
Data 0,1,1,1
'
' Lettera [A]
'
Data 0,1,0,0
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [B]
'
Data 1,1,0,0
Data 1,0,1,0
Data 1,1,0,0
Data 1,0,1,0
Data 1,1,0,0
Data 0,0,0,0
'
' Lettera [C]
'
Data 0,1,0,0
Data 1,0,1,0
Data 1,0,0,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
' Lettera [D]
'
Data 1,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,0,0
Data 0,0,0,0
'
' Lettera [E]
'
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,1,0
Data 0,0,0,0
'
' Lettera [F]
'
Data 1,1,1,0
Data 1,0,0,0
Data 1,1,0,0
Data 1,0,0,0
Data 1,0,0,0
Data 0,0,0,0
'
' Lettera [G]
'
Data 0,1,0,0
Data 1,0,0,0
Data 1,1,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
' Lettera [H]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [I]
'
Data 1,1,1,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 1,1,1,0
Data 0,0,0,0
'
' Lettera [J]
'
Data 1,1,1,0
Data 0,0,1,0
Data 0,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
' Lettera [K]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [L]
'
Data 1,0,0,0
Data 1,0,0,0
Data 1,0,0,0
Data 1,0,0,0
Data 1,1,1,0
Data 0,0,0,0
'
' Lettera [M]
'
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [N]
'
Data 1,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [O]
'
Data 0,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
' Lettera [P]
'
Data 1,1,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,0,0
Data 1,0,0,0
Data 0,0,0,0
'
' Lettera [Q]
'
Data 0,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,1,0
'
' Lettera [R]
'
Data 1,1,0,0
Data 1,0,1,0
Data 1,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'
' Lettera [S]
'
Data 0,1,0,0
Data 1,0,1,0
Data 0,1,0,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'
```

```
' Lettera [T]
'
Data 1,1,1,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,0,0
'


' Lettera [U]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 0,0,0,0
'

' Lettera [V]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,0,0,0
'

' Lettera [W]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,1,1,0
Data 1,0,1,0
Data 0,0,0,0
'

' Lettera [X]
'
Data 1,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,0,0,0
'

' Lettera [Y]
'
Data 1,0,1,0
Data 1,0,1,0
Data 1,0,1,0
Data 0,1,0,0
Data 0,1,0,0
Data 0,0,0,0
'

' Lettera [Z]
'
Data 1,1,1,0
Data 0,0,1,0
Data 0,1,0,0
Data 1,0,0,0
Data 1,1,1,0
Data 0,0,0,0
'
```

MSX

L'ASSEMBLER

```
0 ' ROUTINE DIMOSTRATIVA
1 ' DELLA LEZIONE DI ASSEMBLER
2 ' PER FARE UN BEEP
3 ' IN TIME-SHARING
4 '
5 ' (C) 25-10-87
6 '
7 ' by SILVERSOFT
8 '
10 FOR N=55000! TO 55043!
20 READ A$:POKE N,VAL("&H"+A$):C
K=CK+VAL("&H"+A$):NEXT
30 READ C:IF C<>CK THEN CLS :PRI
NT"ERRORE NEI DATA : RICONTROLLA
":LIST 100-160
40 DEFUSR=55000!:A=USR(0)
99 '
100 DATA 3E,C3,32,9A,FD,21,EC,D6
110 DATA 22,9B,FD,21,9E,FC,36,00
120 DATA 23,36,00,C9,00,2A,9E,FC
130 DATA 7C,FE,01,C0,7D,FE,F4,C0
140 DATA 21,9E,FC,36,00,23,36,00
150 DATA CD,C0,00,C9
160 DATA 5448
```

```
 1                ;ROUTINE PER BEEP
 2                ; PER IL CORSO DI
 3                ;   ASSEMBLER
 4                ;
 5                ;  (C) 25-10-87
 6                ;
 7                ; by  SILVERSOFT
 8                ;
 9                ;
10                ; EQUATES:
11                ;
12                TIME:       EQU  0FC9EH
13                HOOK:       EQU  0FD9AH
14                JPCD:       EQU  0C3H
```

```
15                      TIM1:       EQU  01H
16                      TIM2:       EQU  0F4H
17                      BEEP:       EQU  0C0H
18                      ;
19                                  ORG  55000
20                                  LOAD 55000
21                      ;
22                      ;SCRIVE NELL' HOOK...
23                      ;IL SALTO ALLA ROUTINE
24                      ;
25 D6D8 3EC3                        LD   A,JPCD
26 D6DA 329AFD                      LD   (HOOK),A
27 D6DD 21ECD6                      LD   HL,CLCK
28 D6E0 229BFD                      LD   (HOOK+1),HL
29 D6E3 219EFC                      LD   HL,TIME
30 D6E6 3600                        LD   (HL),0
31 D6E8 23                          INC  HL
32 D6E9 3600                        LD   (HL),0
33 D6EB C9                          RET
34                      ;
35                      ;VERIFICA IL NIBBLE ALTO...
36                      ;
37 D6EC 2A9EFC          CLCK:       LD   HL,(TIME)
38 D6EF 7C                          LD   A,H
39 D6F0 FE01                        CP   TIM1
40 D6F2 C0                          RET  NZ
41                      ;
42                      ;...E QUELLO BASSO
43                      ;
44 D6F3 7D                          LD   A,L
45 D6F4 FEF4                        CP   TIM2
46 D6F6 C0                          RET  NZ
47                      ;
48                      ; AZZERA TIME
49                      ;
50 D6F7 219EFC                      LD   HL,TIME
51 D6FA 3600                        LD   (HL),0
52 D6FC 23                          INC  HL
53 D6FD 3600                        LD   (HL),0
54                      ;
55                      ; EMETTE IL BEEP
56                      ;
57 D6FF CDC000                      CALL BEEP
58 D702 C9                          RET
59                                  END
```

ZIG ZAG

```
1 ' Zig Zag di Ercole Murano
2 '
10 DEFUSR 0=&H41:DEFUSR 1=&H44
20 COLOR 4,1,1:GOSUB 1210
30 '
40 '   DISEGNO DELL'OMINO
50 '
60 OMO$="U12 L4 U3 R6 U2 H2 U2 E
2 R2 F2 D2 G2 D2 R6 D3 L4 D12 L2
 U5 L2 D5 L2"
70 SCREEN 0:WIDTH 40:KEY OFF:COL
```

```
OR 2,1
80 DIM SEQ$(36),Z(36),W(36),C(36
)
90 ' ████████████████████████
100 '██ DESCRIZIONE DEL GIOCO █
110 '████████████████████████
120 PRINT"QUESTO GIOCO CONSISTE
NEL'INDOVINARE LA SEQUENZA ESATT
A DI UNA SERIE DI  OMINI  COLORA
TI DISPOSTI A PIRAMIDE.":PRINT
130 PRINT"PER IMMETTERE LA SEQUE
NZA USA LE INIZIA-LI DEI TRE COL
ORI OVVERO: LA «A» PER AZ-ZURRO,
 LA «R» PER ROSSO ED INFINE LA «
B» PER BIANCO.":PRINT
140 PRINT"PUOI SCEGLIERE FRA TRE
 DIVERSI LIVELLI  DI DIFFICOLTA'
 NATURALMENTE IN FUNZIONE DELLA
LUNGHEZZA DELLA SEQUENZA DEI COL
O-RI.":PRINT
150 PRINT"ATTIVA IL TASTO 'CAP'
E PREMI SPAZIO"
160 K$=INKEY$:IF K$<>" " THEN 16
0
170 R=0:N=0:X=0:Y=0:X1=0
180 '████████████████████████
190 '██ LIVELLO DI DIFICOLTA' █
200 '████████████████████████
210 CLS
220 LOCATE 2,4:PRINT"SELEZIONA I
L LIVELLO DI DIFFICOLTA'"
230 LOCATE 2,5:PRINT"────────────
──────────────────────────────"
240 LOCATE 7,7:PRINT"1) SEQUENZA
 DI 21 COLORI"
250 LOCATE 7,9:PRINT"2) SEQUENZA
 DI 28 COLORI"
260 LOCATE 7,11:PRINT"3) SEQUENZ
A DI 36 COLORI":PRINT
270 K$=INKEY$:IF VAL(K$)<1 OR VA
L(K$)>3 THEN 270
280 BEEP:D=VAL(K$)
290 X=7:Y=190:X1=95+D*20:M=5+D
300 LOCATE 0,13:PRINT"OSSERVA AT
TENTAMENTE L'ORDINE DEI COLO- RI
 DELLA FIGURA CHE APPARIRA' TRA
BREVE.":FOR P=1 TO 2500:NEXT
310 SCREEN 2:OPEN"GRP:"AS#1
320 U=USR(1)
330 '████████████████████████
340 '█ DISEGNA OMINI E LETTERE█
350 '████████████████████████
360 COLOR 1:LINE(168,0)-(255,192
),14,BF
370 DRAW"BM175,30 C4 S4"+OMO$:PA
INT(176,29),4
380 PSET(175,40),14:PRINT#1,"A"
390 PSET(176,40),14:PRINT#1,"A"
400 DRAW"BM205,30 C6"+OMO$:PAINT
(206,29),6
410 PSET(207,40),14:PRINT#1,"R"
420 PSET(208,40),14:PRINT#1,"R"
430 DRAW"BM235,30 C15"+OMO$:PAIN
T(236,29),15
440 PSET(237,40),14:PRINT#1,"B"
450 PSET(238,40),14:PRINT#1,"B"
460 '████████████████████████
470 '██ REALIZZA LA PIRAMIDE ██
480 '████████████████████████
490 FOR S=1 TO M
500 FOR V=X TO X1 STEP 21
510 CO=INT(RND(-TIME)*16)
520 IF CO<6 THEN CO=4
530 IF CO>10 THEN CO=15
540 IF CO=6 OR CO=7 OR CO=8 OR C
O=9 OR CO=10 THEN CO=6
550 DRAW"BM=V;,=Y;S4 C=CO;"+OMO$
560 PAINT(V+1,Y-1),CO
570 GOSUB 620
580 NEXT V
590 X=X+10:Y=Y-23:X1=X1-10
600 NEXT S:U=USR 1(0):GOTO 690
610 '████████████████████████
620 '██ MEMORIZZA LA SEQUENZA █
630 '████████████████████████
640 N=N+1:R=R+1
650 Z(N)=V
660 W(N)=Y
670 C(N)=CO
680 RETURN
690 FOR P=1 TO 30000:NEXT
700 GOSUB 1020
710 N=0
720 '████████████████████████
730 '██ IMMIS. DELLA SEQUENZA █
740 '████████████████████████
750 K$=INKEY$:IF K$="A" OR K$="R
" OR K$="B" THEN GOSUB 760 ELSE
750
760 IF K$="A"THEN K$="4"
770 IF K$="R"THEN K$="6"
780 IF K$="B"THEN K$="15"
790 CC=VAL(K$)
800 LINE(192,50)-(223,90),14,BF
810 DRAW"BM201,85 S6 C=CC;"+OMO$
820 PAINT(202,80),VAL(K$)
830 N=N+1:IF VAL(K$)<>C(N) THEN
890 ELSE DRAW"BM=Z(N);,=W(N);S4
C=C(N);"+OMO$:PAINT(Z(N)+1,W(N)-
1),C(N)
```

```
840 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
850 '▓ CONTROLLO SEQ. IMMESSA ▓
860 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
870 IF VAL(K$)=C(N) AND N=R THEN
 1100
880 PLAY"t255 s8 m58000 v15 o4 l
64 eo5go6dgo":GOTO 750
890 COLOR 6:PSET(180,90),14:PRIN
T#1,"SEQUENZA":PSET(181,90):PRIN
T#1,"SEQUENZA"
900 PSET(186,100),14:PRINT#1,"ER
RATA"
910 PSET(187,100):PRINT#1,"ERRAT
A"
920 PLAY "s8 m18000 t120 o2 A8"
930 DRAW"BM201,155 S6 C=C(N);"+O
MO$
940 PAINT(202,150),C(N)
950 COLOR 1:PSET(180,170),14:PRI
NT#1,"SEQUENZA":PSET(181,170):PR
INT#1,"SEQUENZA"
960 PSET(186,180),14:PRINT#1,"ES
ATTA"
970 PSET(187,180):PRINT#1,"ESATT
A"
980 FOR P=1 TO 500:NEXT:GOSUB 10
20:GOSUB 1060:GOTO 710
990 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1000 '    CANCELLA LA PIRAMIDE
1010 '
1020 LINE(0,0)-(164,192),1,BF:RE
TURN
1030 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1040 'CANC. GLI OMINI DI DESTRA
1050 '
1060 LINE(170,50)-(255,192),14,B
F:RETURN

1070 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1080 '▓    NUOVA PARTITA ?    ▓
1090 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1100 PLAY"V15 S0 M5000 O5 F5 F16
 F16 F4 R4 G8 F8 E8 F4 F16 F16 F
4"
1110 PSET(170,150),14:COLOR 4:PR
INT#1,"BRAVO!!!!":FOR P=1 TO 200
0:NEXT:CLOSE
1120 SCREEN 0:PRINT"VUOI FARE UN
'ALTRA PARTITA?"
1130 K$=INKEY$:IF K$="S"THEN 170
1140 IF K$="N" THEN 1160
1150 IF K$="" THEN 1130
1160 CLS:END
1170 IF CO=15 OR CO=6 OR CO=4 TH
EN 550 ELSE 510
1180 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1190 '▓ PRESENTAZIONE INIZIALE▓
1200 '▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1210 SCREEN 2:X=USR(1)
1220 Z$="R40 G40 R35 G10 L45 E40
 L30 E10"
1222 G$="R40 D10 L30 D30 R20 U10
 L10 U10 R20 D30 L40 U50"
1224 A$="BF10 R20 D10 L20 U10 BH
10 R40 D50 L10 U20 L20 D20 L10 U
50"
1230 DRAW"BM25,10 S3 C4"+Z$:PAIN
T(26,11),4
1240 DRAW"BM75,10 S3 C6 R10 D50
L10 U50":PAINT(76,11),6
1242 DRAW"BM95,10 S3 C15"+G$:PAI
NT(97,13),15
1244 DRAW"BM100,60 S3 C4"+Z$:PAI
NT(101,61),4
1246 DRAW"BM150,60 C6"+A$:PAINT(
152,62),6
1248 DRAW"BM200,60 C15"+G$:PAINT
(202,62),15
1250 OPEN"GRP:"AS#1
1260 PRESET(18,120),1:PRINT#1,"V
ERSIONE PER COMPUTER MSX"
1270 PRESET(19,120),1:PRINT#1,"V
ERSIONE PER COMPUTER MSX"
1280 PRESET(6,129),1:PRINT#1,"DE
LL'OMONIMO GIOCO TELEVISIVO"
1290 PRESET(5,129),1:PRINT#1,"DE
LL'OMONIMO GIOCO TELEVISIVO"
1300 PRESET(5,170),1:PRINT#1," (
C) 1987 BY MURANO ERCOLE"
1310 PRESET(6,170),1:COLOR 15:PR
INT#1," (C) 1987 BY MURANO ERCOL
E"
1320 X=USR 1(0):FOR T = 1 TO 200
0:NEXT:CLOSE:RETURN
```

64 COLONNE IN SCREEN 2

```
0 ' ROUTINE PER INPUT
1 '   A 64 COLONNE
2 '
3 '   (C) 25-10-87
4 '
5 '  by  SILVERSOFT
6 '
10 COLOR 14,1,1
```

```
17 '
18 ' LEGGE PATTERN
19 '
20 CLEAR 500,54499!:RESTORE 1000
0:CK=0
30 FOR N=54500! TO 54755!
40 READ A$:POKE N,VAL("&H"+A$)
50 CK=CK+VAL("&H"+A$)
60 NEXT
70 READ CC:IF CC<>CK THEN PRINT"
ERRORE NEI DATA : RICONTROLLA":S
TOP
117 '
118 ' LEGGE L/M
119 '
120 RESTORE 20000:CK=0
130 FOR N=54800! TO 54990!
140 READ A$:POKE N,VAL("&H"+A$)
150 CK=CK+VAL("&H"+A$)
160 NEXT
170 READ CC:IF CC<>CK THEN PRINT
"ERRORE NEI DATA : RICONTROLLA":
STOP
180 DEFUSR=54800!
497 '
498 ' ESEMPIO DI UTILIZZO
499 '
500 COLOR 15,4,7:SCREEN 2:CLS
510 PRESET(32,16):P$=USR("PROVA
DI UTILIZZO ROUTINE 64 COLONNE I
N SCREEN 2")
520 X=80:Y=40:GOSUB 1000
530 END
997 '
998 ' INPUT CON 64 COLONNE
999 '
1000 DEF FNP(A$)=ABS(2-1*(LEN(A$
)-(LEN(A$)\2)*2)):DEFUSR=54800!:
PRESET(X,Y):A$="":X=X+4:P$=USR("
X=# ")
1010 K$=INPUT$(1):K=ASC(K$):IF K
=13 THEN RETURN ELSE IF K=8 AND
A$<>"" THEN A$=LEFT$(A$,LEN(A$)-
1):X=X-4:GOTO 1030 ELSE IF(K<32
OR K>95) OR LEN(A$)>200 THEN 101
0
1020 A$=A$+K$:X=X+4
1030 IF X<88 AND Y=40 THEN X=80:
GOTO 1000
1035 IF X<0 THEN X=252:Y=Y-8
1036 IF X>252 THEN X=0:Y=Y+8
1040 PRESET(X,Y)
1050 P$=USR(RIGHT$(A$,FNP(A$))+"
#    ")
1060 GOTO 1010
```

```
10000 '
10010 ' DATA PER I PATTERNS
10020 '
10030 DATA 00,00,00,00,00,44,40,
40
10040 DATA 00,AA,00,00,00,FF,FF,
F0
10050 DATA 00,EC,E6,E0,00,A2,48,
A0
10060 DATA 00,4A,C8,60,00,48,00,
00
10070 DATA 00,48,88,40,00,84,44,
80
10080 DATA 00,4E,4E,40,00,44,E4,
40
10090 DATA 00,00,00,48,00,00,E0,
00
10100 DATA 00,00,04,40,00,22,48,
80
10110 DATA 00,EA,AA,E0,00,4C,44,
E0
10120 DATA 00,E2,E8,E0,00,E2,62,
E0
10130 DATA 00,88,AE,20,00,E8,E2,
E0
10140 DATA 00,E8,EA,E0,00,E2,44,
40
10150 DATA 00,EA,EA,E0,00,EA,E2,
E0
10160 DATA 00,04,04,00,00,00,40,
48
10170 DATA 00,24,84,20,00,0E,0E,
00
10180 DATA 00,84,24,80,00,E2,E0,
80
10190 DATA 00,C2,2A,C0,00,EA,EA,
A0
10200 DATA 00,EA,CA,E0,00,E8,88,
E0
10210 DATA 00,CA,AA,C0,00,E8,C8,
E0
10220 DATA 00,E8,C8,80,00,E8,AA,
E0
10230 DATA 00,AA,EA,A0,00,44,44,
40
10240 DATA 00,22,2A,E0,00,AA,CA,
A0
10250 DATA 00,88,88,E0,00,AE,AA,
A0
10260 DATA 00,CA,AA,A0,00,EA,AA,
E0
10270 DATA 00,EA,E8,80,00,EA,AE,
E0
10280 DATA 00,EA,EC,A0,00,E8,E2,
E0
```

```
10290 DATA 00,E4,44,40,00,AA,AA,
E0
10300 DATA 00,AA,AA,40,00,AA,AE,
A0
10310 DATA 00,AA,4A,A0,00,AA,44,
40
10320 DATA 00,E2,48,E0,00,C8,88,
C0
10330 DATA 00,88,42,20,00,C4,44,
C0
10340 DATA 00,4A,00,00,00,00,00,
E0
10350 DATA 26924
20000 '
20010 ' DATA PER LA ROUTINE L/M
20020 '
20030 DATA 3A,B9,FC,E6
20040 DATA F8,6F,26,00,29,29,3A,
B7
20050 DATA FC,CB,3F,CB,3F,CB,3F,
16
20060 DATA 00,5F,19,29,29,29,E5,
2A
20070 DATA F8,F7,7E,32,08,D6,23,
5E
20080 DATA 23,56,ED,53,0A,D6,DD,
2A
20090 DATA 0A,D6,DD,2B,0E,00,DD,
23
20100 DATA E1,3A,08,D6,B9,C8,0C,
E5
20110 DATA DD,7E,00,D6,20,6F,E6,
C0
20120 DATA C2,5A,47,26,00,29,29,
11
20130 DATA E4,D4,19,22,0C,D6,DD,
23
20140 DATA E1,3A,08,D6,B9,C8,0C,
E5
20150 DATA DD,7E,00,D6,20,6F,E6,
C0
20160 DATA C2,5A,47,26,00,29,29,
11
20170 DATA E4,D4,19,22,0E,D6,06,
04
20180 DATA 2A,0C,D6,7E,E6,F0,57,
2A
20190 DATA 0E,D6,7E,E6,F0,CB,3F,
CB
20200 DATA 3F,CB,3F,CB,3F,B2,E1,
CD
20210 DATA 4D,00,23,E5,2A,0C,D6,
7E
20220 DATA E6,0F,CB,27,CB,27,CB,
27
20230 DATA CB,27,57,2A,0E,D6,7E,
E6
20240 DATA 0F,B2,E1,CD,4D,00,23,
E5
20250 DATA 2A,0C,D6,23,22,0C,D6,
2A
20260 DATA 0E,D6,23,22,0E,D6,10,
B8
20270 DATA C3,42,D6
20280 DATA 21874
```

```
1                ; ROUTINE PER SCREEN
2                ;     A 64 COLONNE
3                ;      (C) 12/8/87
4                ;     by SILVERSOFT
5                ;
6                ;
7                ; EQUATES:
8                ;
9                COGRX:      EQU  0FCB7H
10               COGRY:      EQU  0FCB9H
11               PATTN:      EQU  54500
12               PNTT1:      EQU  54796
13               PNTT2:      EQU  54798
14               PNTUS:      EQU  0F7F8H
15               DSCUS:      EQU  54794
16               LNSTR:      EQU  54792
17               FCERR:      EQU  0475AH
18
19                           LOAD 54800
20                           ORG  54800
```

```
21                      ;
22                      ;CALCOLA L'INDIRIZZO
23                      ; SULLO SCHERMO
24                      ;
25 D610 3AB9FC                  LD   A,(COGRY)
26 D613 E6F8                    AND  0F8H
27 D615 6F                      LD   L,A
28 D616 2600                    LD   H,0
29 D618 29                      ADD  HL,HL
30 D619 29                      ADD  HL,HL
31 D61A 3AB7FC                  LD   A,(COGRX)
32 D61D CB3F                    SRL  A
33 D61F CB3F                    SRL  A
34 D621 CB3F                    SRL  A
35 D623 1600                    LD   D,0
36 D625 5F                      LD   E,A
37 D626 19                      ADD  HL,DE
38 D627 29                      ADD  HL,HL
39 D628 29                      ADD  HL,HL
40 D629 29                      ADD  HL,HL
41 D62A E5                      PUSH HL
42                      ;
43                      ;CARICA IN IX
44                      ;L'INIZIO DELLA
45                      ;STRINGA
46                      ;
47 D62B 2AF8F7                  LD   HL,(PNTUS)
48 D62E 7E                      LD   A,(HL)
49 D62F 3208D6                  LD   (LNSTR),A
50 D632 23                      INC  HL
51 D633 5E                      LD   E,(HL)
52 D634 23                      INC  HL
53 D635 56                      LD   D,(HL)
54 D636 ED530AD6                LD   (DSCUS),DE
55 D63A DD2A0AD6                LD   IX,(DSCUS)
56 D63E DD2B                    DEC  IX
57 D640 0E00                    LD   C,0
58 D642 DD23            CALP:   INC  IX
59                      ;
60                      ;CONTROLLA LA FINE
61                      ;
62 D644 E1                      POP  HL
63 D645 3A08D6                  LD   A,(LNSTR)
64 D648 B9                      CP   C
65 D649 C8                      RET  Z
66 D64A 0C                      INC  C
67 D64B E5                      PUSH HL
68                      ;
69                      ;CALCOLA PUNTATORE
70                      ;DELLA PRIMA TABELLA
71                      ;
72 D64C DD7E00                  LD   A,(IX+0)
73 D64F D620                    SUB  20H
74 D651 6F                      LD   L,A
75 D652 E6C0                    AND  192
```

```
 76 D654 C25A47                 JP   NZ,FCERR
 77 D657 2600                   LD   H,0
 78 D659 29                     ADD  HL,HL
 79 D65A 29                     ADD  HL,HL
 80 D65B 11E4D4                 LD   DE,PATTN
 81 D65E 19                     ADD  HL,DE
 82 D65F 220CD6                 LD   (PNTT1),HL
 83 D662 DD23                   INC  IX
 84                     ;
 85                     ;CONTROLLA LA FINE
 86                     ;
 87 D664 E1                     POP  HL
 88 D665 3A08D6                 LD   A,(LNSTR)
 89 D668 B9                     CP   C
 90 D669 C8                     RET  Z
 91 D66A 0C                     INC  C
 92 D66B E5                     PUSH HL
 93                     ;
 94                     ;CALCOLA PUNTATORE
 95                     ;SECONDA TABELLA
 96                     ;
 97 D66C DD7E00                 LD   A,(IX+0)
 98 D66F D620                   SUB  20H
 99 D671 6F                     LD   L,A
100 D672 E6C0                   AND  192
101 D674 C25A47                 JP   NZ,FCERR
102 D677 2600                   LD   H,0
103 D679 29                     ADD  HL,HL
104 D67A 29                     ADD  HL,HL
105 D67B 11E4D4                 LD   DE,PATTN
106 D67E 19                     ADD  HL,DE
107 D67F 220ED6                 LD   (PNTT2),HL
108                    ;
109                    ;SCRIVE GLI OTTO
110                    ;BYTES DEL CARATTERE
111                    ;
112 D682 0604                   LD   B,4
113 D684 2A0CD6        WCAR:    LD   HL,(PNTT1)
114 D687 7E                     LD   A,(HL)
115 D688 E6F0                   AND  240
116 D68A 57                     LD   D,A
117 D68B 2A0ED6                 LD   HL,(PNTT2)
118 D68E 7E                     LD   A,(HL)
119 D68F E6F0                   AND  240
120 D691 CB3F                   SRL  A
121 D693 CB3F                   SRL  A
122 D695 CB3F                   SRL  A
123 D697 CB3F                   SRL  A
124 D699 B2                     OR   D
125 D69A E1                     POP  HL
126 D69B CD4D00                 CALL 004DH
127 D69E 23                     INC  HL
128 D69F E5                     PUSH HL
129 D6A0 2A0CD6                 LD   HL,(PNTT1)
130 D6A3 7E                     LD   A,(HL)
```

```
131 D6A4 E60F        AND   15
132 D6A6 CB27        SLA   A
133 D6A8 CB27        SLA   A
134 D6AA CB27        SLA   A
135 D6AC CB27        SLA   A
136 D6AE 57          LD    D,A
137 D6AF 2A0ED6      LD    HL,(PNTT2)
138 D6B2 7E          LD    A,(HL)
139 D6B3 E60F        AND   15
140 D6B5 B2          OR    D
141 D6B6 E1          POP   HL
142 D6B7 CD4D00      CALL  004DH
143 D6BA 23          INC   HL
144 D6BB E5          PUSH  HL
145 D6BC 2A0CD6      LD    HL,(PNTT1)
146 D6BF 23          INC   HL
147 D6C0 220CD6      LD    (PNTT1),HL
148 D6C3 2A0ED6      LD    HL,(PNTT2)
149 D6C6 23          INC   HL
150 D6C7 220ED6      LD    (PNTT2),HL
151 D6CA 10B8        DJNZ  WCAR
152 D6CC C342D6      JP    CALP
153                  END
```

AUTOLOADER

```
10 ' ┌──────────────────────────────┐
20 ' │                              │
30 ' │     MILANO  02/1987          │
40 ' │                              │
50 ' │  Autoload  Automatico        │
60 ' │                              │
70 ' │  (C) Gianni Bellomusto       │
80 ' │                              │
90 ' └──────────────────────────────┘
100 ' ── inizializza sistema ──
110 CLEAR 2000:SCREEN 0:CLS:ON E
RROR GOTO 890:GOSUB 1020
120 ON KEY GOSUB 820,840:MX=PEEK
(&H2D):IF MX=1 THEN MS=38:A=2:WI
DTH 80 ELSE WIDTH 40:MS=18:A=1
130 T$=DSKI$(1,0):F=PEEK(&HF352)
*256+PEEK(&HF351):FD=PEEK(F+19)+
PEEK(F+20)*256:VA$=STRING$(32,"
"):LO=VARPTR(VA$)
140 IF FD>720 THEN SE=7ELSE SE=5
150 P1=MS-8-(A*7):P2=MS+(A*7):T=
1:YP=3
160 Y1=1:YS=YP:P=P1:PP=P1
170 DIM A$(120)
180 COLOR 15,1:KEY OFF:Z=0
190 ' ── lettura dischi ──
200 CLS:LOCATE MS-10,11:PRINT"le
ggo la directory...."
210 GOSUB 1020
220 FOR Y=SE TO SE+6
230  DS$=DSKI$(1,Y)
240 ' inserisce files in vettore
250   FOR D=0 TO 15
260    DO=F+D*32
270    IF PEEK(DO)=229 OR PEEK(D
O)=0 THEN 320
280    Z=Z+1
290    POKE LO+2,DO/256:POKE LO+
1,DO-PEEK(LO+2)*256
300    A$(Z)=LEFT$(VA$,11)
310   A$(Z+1)=""
320  NEXT D
330 NEXT Y
340 '──────────────────────────
350 '  stampa videata
360 '──────────────────────────
370 KEY(1)ON:KEY(2)ON
380 CLS:LOCATE MS-4,23:PRINT"Qua
le scelta";
390 LOCATE MS-8,0:PRINT"AUTOLOAD
 AUTOMATICO"
400 LOCATE 0,3:FOR Y=Y1 TO Y1+19
 STEP 2
410 PRINT TAB(P1);A$(Y);:PRINT T
AB(P2);A$(Y+1):PRINT
420 NEXT
430 LOCATE PP-1,YP-1:PRINT"
```

```
          "
440 LOCATE PP-1,YP:PRINT"
          "
450 LOCATE PP-1,YP+1:PRINT"
          "
460 LOCATE PP,YP:PRINT B$
470 LOCATE P-1,YS-1:PRINT"┌──────
──────┐"
480 LOCATE P-1,YS:PRINT"|";A$(T)
490 LOCATE P-1+12,YS:PRINT"|"
500 LOCATE P-1,YS+1:PRINT"└──────
──────┘"
510 '──────────────────────────────
520 '  gestione della scelta
530 '──────────────────────────────
540 Y$=INKEY$:IF Y$=""THEN 540
550 W=ASC(Y$):IF MX<>1 THEN 600
560 IF W=28 AND T=Z THEN 640 ELS
E IF W=28 AND P=52 THEN 540 ELSE
 IF W=28 THEN YP=YS:B$=A$(T):T=T
+1:PP=P:P=P2:GOTO 430
570 IF W=29 AND P=16 THEN 540 EL
SE IF W=29 THEN YP=YS:B$=A$(T):T
=T-1:PP=P:P=P1:GOTO 430
580 IF(W=30 AND YS<4)THEN 540 EL
SE IF W=30 THEN B$=A$(T):T=T-2:Y
P=YS:YS=YS-2:PP=P:GOTO 430
590 IF T=Z OR T=Z-1 THEN 640 ELS
E IF(W=31 AND YS>20)THEN 540 ELS
E IF W=31 THEN B$=A$(T):T=T+2:YP
=YS:YS=YS+2:PP=P:GOTO 430
600 IF W=28 AND T=Z THEN 640 ELS
E IF W=28 AND P=25 THEN 540 ELSE
 IF W=28 THEN YP=YS:B$=A$(T):T=T
+1:PP=P:P=P2:GOTO 430
610 IF W=29 AND P=3 THEN 540 ELS
E IF W=29 THEN YP=YS:B$=A$(T):T=
T-1:PP=P:P=P1:GOTO 430
620 IF(W=30 AND YS<4)THEN 540 EL
SE IF W=30 THEN PP=P:B$=A$(T):T=
T-2:YP=YS:YS=YS-2:GOTO 430
630 IF T=Z OR T=Z-1 THEN 640 ELS
E IF(W=31 AND YS>20)THEN 540 ELS
E IF W=31 THEN PP=P:B$=A$(T):T=T
+2:YP=YS:YS=YS+2:GOTO 430
640 IF W<>13 THEN 430
650 FR$="ren msxdos.sys msxdos"
660 EX$=RIGHT$(A$(T),3)
670 IF EX$="COM"THEN 690
680 IF EX$="BAT"THEN 700
685 GOTO 710
690 NAME"MSXDOS"AS"MSXDOS.SYS":O
PEN"AUTOEXEC.BAT"FOR OUTPUT AS#1
:PRINT#1,FR$:PRINT#1,LEFT$(A$(T)
,8):CLOSE#1:GOSUB 1020:DEFUSR=0:
A=USR(0)
700 NAME"MSXDOS"AS"MSXDOS.SYS":O
PEN"AUTOEXEC.BAT"FOR OUTPUT AS#1
:PRINT#1,FR$:PRINT#1,LEFT$(A$(T)
,8):CLOSE#1:GOSUB 1020:DEFUSR=0:
A=USR(0)
710 GOSUB 960
720 IF VA<>255 AND VA<>254 THEN
ERROR 255
730 IF VA=255 THEN 790
740 ' -- LINGUAGGIO MACCHINA --
750 IF VA=254 AND I=0 THEN CLS:L
OCATE MS-10,11:PRINT A$(T);" ";"
IS LOADING...":FOR TT=1 TO 800:N
EXT:SCREEN 2:BLOAD A$(T),S
760 GOSUB 1020:RUN
770 CLS:LOCATE MS-10,11:PRINT A$
(T);" ";"IS LOADING...":BLOAD A$
(T),R
780 ' -- BASIC --
790 CLS:LOCATE MS-10,11:PRINT A$
(T);" ";"IS LOADING...":RUN A$(T
)
800 ' ------------------------
810 FOR TT=1 TO 500:NEXT:RETURN
820 IF Y1=<1 THEN Y1=1:RETURN 54
0
830 FA=FA-1:Y1=Y1-20:T=T-20:YP=3
:YS=3:P=P1:PP=P1:B$=A$(T):IF T>2
0 THEN T=T-(T-1)+20*FA:RETURN 38
0 ELSE IF T<20 THEN T=1:RETURN 3
80
840 IF Y>=Z THEN RETURN 540
850 FA=FA+1:Y1=Y1+20:T=T+20:YP=3
:YS=3:P=P1:PP=P1:B$=A$(T):IF T>2
0 THEN T=T-(T-1)+20*FA:RETURN 38
0 ELSE IF T<20 THEN T=1
860 '──────────────────────────────
870 '  gestione degli errori
880 '──────────────────────────────
890 IF ERR=66 OR ERR=69 OR ERR=7
0 OR ERR=68 OR ERR=67 THEN SN$="
ERRORE DI DISCO ! VERIFICA E PRE
MI UN TASTO":CLS:LOCATE MS-18,11
:PRINT SN$;Y$=INPUT $(1):IF EX$=
"COM"THEN RESUME 690 ELSE IF EX$
="BAT"THEN RESUME 700
900 IF ERR=255 THEN SN$="PROGRAM
MA NON ESEGUIBILE!":GOTO 910
910 CLS:LOCATE MS-10,11:PRINT SN
$
920 FOR TT=1 TO 2000:NEXT:RESUME
 380
930 '──────────────────────────────
940 ' controllo tipo programma
```

```
950 '------------------------------
960 OPEN A$(T) AS#1
970 FIELD#1,7 AS F$
980 GET#1,1
990 VA=ASC(MID$(F$,1,1))
1000 I=ASC(MID$(F$,2,1))+256*ASC
(MID$(F$,3,1))
1010 CLOSE#1
1020 FOR TT=1 TO 2000:NEXT:RETUR
N
1030 '------------------------------
```

MUTUI

```
1000 '┌──────────────┐
1010 '|  MSX MUTUI   |
1020 '|FRANKYSOFT 86|
1030 '|VERSIONE 2.03|
1040 '|  (c)  1987   |
1050 '└──────────────┘
1060 SCREEN 0
1070 WIDTH 40
1080 KEY OFF
1090 X$="------------------------------"
1100 X$=X$+X$
1110 F$=" ############"
1120 L$=STRING$(40,195)
1130 R$="Ratei "
1140 DEFUSR=&H156
1150 DIM A$(12)
1160 X1=15
1170 X2=1
1180 L=1
1190 GOSUB 2220
1200 RESTORE 2600
1210 FOR Q=1 TO 10
1220  READ A$
1230  LOCATE(40-LEN(A$))\2,7+Q
1240  PRINT A$
1250 NEXT
1260 GOSUB 1910
1270 COLOR 15,4,4
1280 W=1
1290 CLS
1300 GOSUB 2220
1310 ON W GOSUB 1710,1720,1730,1
740,1750,1760
1320 CLS
1330 L=1
1340 GOSUB 2220
1350 LOCATE 10,4
1360 PRINT"Piano di ammortamento
"
1370 LOCATE 1
1380 PRINT"Numero rate__"USING"
##";N
1390 LOCATE 25,5
1400 PRINT"Tasso__"USING" ###";I
*100;
1410 LOCATE 1
1420 PRINT"Importo rata_"INT(K)
1430 LOCATE 1
1440 PRINT"Capitale_____________
__"USING F$;C
1450 LOCATE 1
1460 PRINT"Interessi____________
__"USING F$;H
1470 LOCATE 1
1480 PRINT"Totale da rimborsare_
__"USING F$;Z
1490 O=10
1500 FOR Q=1 TO N STEP 10
1510  U=Q+9
1520  FOR J=Q TO U
1530   O=O+1
1540   A=Z-(K*J)
1550   LOCATE 1,O
1560   PRINT"Saldo dopo rata n°
"USING"##  ";J;
1570   PRINT USING F$;A
1580   IF A<=10 THEN 1670
1590  NEXT
1600  O=10
1610  LOCATE 14,22
1620  PRINT"Premi Spazio"
1630  IF STRIG(0)<>-1 THEN 1610
1640  LOCATE 0,22
1650  PRINT SPC(39)
1660 NEXT
1670 IF O<20 AND N<>10 AND N<>20
 AND N<>30 THEN FOR QQ=1 TO 20-O
:O=O+1:LOCATE 1,O:PRINT SPC(38):
NEXT
1680 LOCATE 12,22
1690 PRINT"Spazio per Menù"
1700 IF STRIG(0)THEN 1270 ELSE 1
680
1710 PRINT R$ A$ (7):GOSUB 1990:
RETURN
1720 PRINT R$ A$ (8):GOSUB 1990:
RETURN
1730 PRINT R$ A$ (9):GOSUB 1990:
RETURN
1740 PRINT R$ A$(10):GOSUB 1990:
RETURN
1750 PRINT R$ A$(11):GOSUB 1990:
RETURN
1760 FOR Q=5 TO 19
1770  LOCATE 3,Q
```

```
1780  PRINT SPC(35)
1790 NEXT
1800 RESTORE 2540
1810 FOR I=1 TO 11
1820  READ A$
1830  LOCATE(40-LEN(A$))\2,5+I
1840  PRINT A$
1850 NEXT
1860 LOCATE,22
1870 END
1880 '
1890 'PARTI COMUNI
1900 '
1910 LOCATE 14,22
1920 PRINT"Premi Spazio"
1930 IF STRIG(0)THEN LOCATE 0,22
:PRINT SPC(39):RETURN
1940 COLOR X1
1950 FOR Q=1 TO 200
1960 NEXT
1970 SWAP X1,X2
1980 GOTO 1910
1990 PRINT L$
2000 N=0
2010 PRINT"0 > Esci"
2020 LOCATE,5
2030 INPUT"N.ro rate___";N
2040 IF N<=0 THEN 1270
2050 INPUT"Capitale____";C
2060 INPUT"Tasso_______";I
2070 IF W=1 THEN M=1
2080 IF W=2 THEN M=12
2090 IF W=3 THEN M=6
2100 IF W=4 THEN M=3
2110 IF W=5 THEN M=2
2120 I=I/100
2130 D=(I/M)*((1+(I/M))^N)
2140 E=((1+(I/M))^N)-1
2150 K=C*(D/E)
2160 Z=(K*N)
2170 H=(K*N)-C
2180 RETURN
2190 '
2200 'Menù
2210 '
2220 PRINT"┌"X$"┐|"SPC(10)"Msx C
omputer System"SPC(9)"||      (c
) 1987 by FrankySoft 86"SPC(7)"|
├"X$"┤";
2230 FOR Q=1 TO 17
2240  PRINT"|"SPC(38)"|";
2250 NEXT
2260 PRINT"└"X$"┘"
2270 IF L THEN L=0:RETURN
2280 RESTORE 2470
2290 FOR Q=1 TO 12
2300  READ A$(Q)
2310  LOCATE(40-LEN(A$(Q)))\2,5+
Q
2320  PRINT A$(Q)
2330 NEXT
2340 IF W<1 THEN W=6
2350 IF W>6 THEN W=1
2360 LOCATE 11,11+W
2370 PRINT"»"
2380 A=STICK(0)
2390 FOR R=1 TO 50
2400 NEXT
2410 IF A=1 THEN LOCATE 11,11+W:
PRINT" ":W=W-1
2420 IF A=5 THEN LOCATE 11,11+W:
PRINT" ":W=W+1
2430 IF STRIG(0)THEN B=USR(0):IF
 W=6 THEN RETURNELSE CLS:RETURNE
LSE 2340
2440 '
2450 'TIPI DI RATEI
2460 '
2470 DATA Ammortamento Mutui,
2480 DATA Menù Generale,,Ratei a
mmessi,
2490 DATA Annuali,Mensili,Bimest
rali
2500 DATA Quadrimestrali,Semestr
ali,Fine
2510 '
2520 'FINE
2530 '
2540 DATA Ciao !,,,,Ammortamento
 Mutui,
2550 DATA by FrankySoft 86,
2560 DATA Versione 2.03,,(c) 198
7
2570 '
2580 'SPIEGAZIONI
2590 '
2600 DATA MUTUI,- - - - -,,Quest
o programma
2610 DATA calcola la RATA COSTAN
TE
2620 DATA da pagare per estingue
re
2630 DATA un dato debito con
2640 DATA i relativi interessi,
2650 DATA Versione 2.03
```

QUIZ

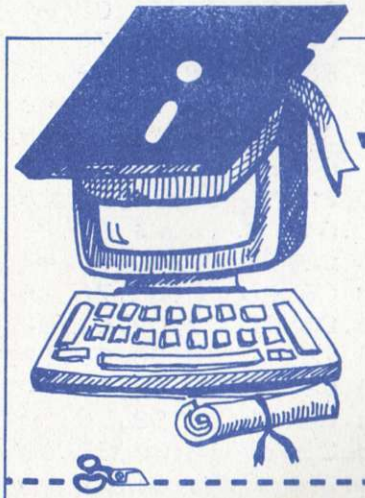
```
10 REM ******** QUIZ *********
20 REM *                     *
30 REM *      VERSIONE 1.7   *
40 REM *                     *
50 REM *          BY         *
60 REM * FRANCESCO MARTINELLI*
70 REM *                     *
80 REM *   TEL.0575/320144   *
90 REM ***********************
100 REM
110 SCREEN 0:WIDTH 40:KEY OFF:CL
S:COLOR 15,1,1:YB=40
120 OUT &HAA,INP(&HAA) OR 64 XOR
 64:POKE &HFCAB,255
130 REM
140 REM Inserimento livello
150 REM
160 CLS:LOCATE 0,21:INPUT"LIVELL
O (MAX 10) ";LIV
170 IF LIV>10 OR LIV<1 THEN GOTO
 160
180 IF LIV=1 THEN LIVE=1:GOTO 23
0
190 LIVE=LIV*25
200 REM
210 REM Inserisci parola ?
220 REM
230 CLS:LOCATE 0,21:INPUT "VUOI
INSERIRE LA PAROLA (S/N) ";A$
231 IF A$="" THEN CLS:GOTO 230
240 IF A$="S" THEN CLS:LOCATE 0,
21:INPUT "PAROLA ";PA$:GOTO 330
250 REM
260 REM MSX sceglie parola
270 REM
280 RN=INT(RND(-TIME)*100)
290 IF RN>35 THEN GOTO 280
300 RESTORE 1080:FOR N=1 TO RN
310 READ PA$
320 NEXT
330 BEEP:CLS
340 LOCATE 0,12:PRINT"PREMI "1"
PER INIZIARE"
350 J$=INKEY$
360 IF J$="1" THEN GOTO 410
370 GOTO 350
380 REM
390 REM Stampa tabellone
400 REM
410 SCREEN 3
420 OPEN "GRP:" AS#1
430 PSET (0,40),1:PRINT#1," ABCD
EFG HIJKLMN OPQRSTU VWXYZ,."
450 COLOR 3
460 PSET(70,0),1:PRINT#1,"QUIZ"
480 REM
490 ' Ricerca coordinate lettere
500 REM
510 FOR N=1 TO LEN (PA$):YA=0:YB
=0
520 RESTORE 860
530 FOR F=1 TO 28
540 READ G$:READ YA:READ YB
550 IF G$=(MID$(PA$,N,1)) THEN G
OTO 570
560 NEXT F
570 COLOR 15
580 REM
590 'Pausa a seconda del livello
600 REM
610 FOR G=1 TO LIVE:NEXT G
620 REM
630 REM Evidenzia la lettera
640 REM
650 PSET (YA,YB),1:PRINT#1,"■"
660 COLOR 1
670 PSET (YA,YB),1:PRINT#1,"■"
680 COLOR 15
690 PSET (YA,YB),1:PRINT#1,G$
700 NEXT N
710 PLAY"DGFBCD"
720 SCREEN 0
730 LOCATE 0,20:PRINT"QUALE ERA
LA PAROLA":INPUT"MISTERIOSA ";PB
$
731 IF PB$="" THEN CLS:GOTO 730
740 REM
750 REM Routine parola errata
760 REM
770 IF PB$<>PA$ THEN PLAY "CCDF"
:LOCATE 0,12:PRINT"HAI SBAGLIATO
 MI DISPIACE LA PAROLA ERA:";PA$
:GOTO  1010
780 REM
790 REM  Routine parola esatta
800 REM
810 IF PB$=PA$ THEN PLAY "DFCC":
LOCATE 0,12:PRINT"HAI INDOVINATO
  BRAVO!!!!!":GOTO  1010
820 COLOR 15,4,4:GOTO 820
830 REM
840 REM Lettere dell'alfabeto
850 REM
860 DATA "A",32,40,"B",64,40,"C"
,96,40,"D",128,40,"E",160,40,"F"
,192,40,"G",224,40,"H",32,72,"I"
,64,72,"J",96,72,"K",128,72,"L",
```

```
160,72,"M",192,72,"N",224,72,"O"
,32,104,"P",64,104,"Q",96,104,"R
",128,104,"S",160,104,"T",192,10
4,"U",224,104,"V",32,136
861 DATA "W",64,136,"X",96,136,"
Y",128,136,"Z",160,136,",",192,1
36,".",224,136
999 '
1010 LOCATE 0,15:PRINT"PREMI ~1~
 PER RIPARTIRE"
1020 J$=INKEY$
1030 IF J$="1" THEN RUN
1040 GOTO 1020
1050 REM
1060 REM Parole per MSX
1070 REM
1080 DATA "GIOSTRA","CAVALLIERE"
,"CAVALLO","LUPO","CAMALEONTE","
CORNOVAGLIA","PRISMA","TRENO","I
NGLESE","DESTRA","SINISTRA","CAR
IOLA","MACCHINA","ASSASSINO","TO
MBINO","CUBO","TESTONE","VOLPE
1090 DATA "GIOSTRATORE","GIOCOLI
ERE","ACQUEDOTTO","QUADRI","ESPO
SIZIONE","CASA","BICICLETTA","RI
DOTTO","RAZZO","PARTENZA","RICIC
LARE","SUDARE","SPIRITO","FANTAS
MA","PIROMANE","SIGARO","UCCELLO
"
```

USA & ABUSA

```
1 ' Nuovo set di caratteri
2 ' per Screen 0
3 ' by Francesco Scarfato
4 '
27 '
28 ' Forma Caratteri
29 '
32 DATA 00,00,00,00,00,00,00,00
33 DATA 20,60,60,60,40,00,60,00
34 DATA D8,D8,D8,00,00,00,00,00
35 DATA 50,F8,F8,50,F8,F8,50,00
36 DATA 20,F8,C0,F8,18,F8,20,00
37 DATA C0,C8,18,30,60,D8,98,00
38 DATA 40,A0,40,A8,90,98,60,00
39 DATA 30,60,C0,00,00,00,00,00
40 DATA 30,60,C0,C0,C0,60,30,00
41 DATA 60,30,18,18,18,30,60,00
42 DATA A8,F8,70,20,70,F8,A8,00
43 DATA 00,20,20,F8,F8,20,20,00
44 DATA 00,00,00,00,00,60,60,C0
45 DATA 00,00,00,78,78,00,00,00
46 DATA 00,00,00,00,00,60,60,00
47 DATA 00,08,18,30,60,C0,80,00
48 DATA 70,C8,D8,E8,C8,C8,70,00
49 DATA 20,60,E0,E0,60,60,F0,00
50 DATA 70,98,18,30,60,C0,F8,00
51 DATA F8,18,10,30,18,98,70,00
52 DATA 30,50,90,F8,10,10,38,00
53 DATA F8,C0,C0,F0,18,18,F0,00
54 DATA 70,C0,C0,F0,D8,D8,70,00
55 DATA F8,18,18,30,60,60,60,00
56 DATA 70,D8,D8,70,D8,D8,70,00
57 DATA 70,D8,D8,78,18,18,70,00
58 DATA 00,60,60,00,00,60,60,00
59 DATA 00,00,60,60,00,60,60,C0
60 DATA 18,30,60,C0,60,30,18,00
61 DATA 00,F8,F8,00,F8,F8,00,00
62 DATA C0,60,30,18,30,60,C0,00
63 DATA 70,98,18,30,60,00,60,00
64 DATA 78,84,B4,A4,B4,84,78,00
65 DATA 20,50,C8,C8,F8,C8,C8,00
66 DATA F0,68,68,70,68,68,F0,00
67 DATA 30,48,C0,C0,C0,48,30,00
68 DATA E0,70,68,68,68,70,E0,00
69 DATA F8,C0,C0,F0,C0,C0,F8,00
70 DATA F8,C0,C0,F0,C0,C0,C0,00
71 DATA 70,C8,C0,D8,C8,C8,70,00
72 DATA C8,C8,C8,F8,C8,C8,C8,00
73 DATA F0,60,60,60,60,60,F0,00
74 DATA 78,30,30,30,B0,B0,60,00
75 DATA C8,D0,E0,E0,D0,C8,C8,00
76 DATA C0,C0,C0,C0,C0,C0,F8,00
77 DATA C4,EC,D4,D4,C4,C4,C4,00
78 DATA C8,C8,C8,E8,D8,D8,C8,00
79 DATA 70,C8,C8,C8,C8,C8,70,00
80 DATA F0,C8,C8,F0,C0,C0,C0,00
81 DATA 70,C8,C8,C8,C8,D0,68,00
82 DATA F0,C8,C8,F0,E0,D0,C8,00
83 DATA 70,C8,C0,70,18,98,70,00
84 DATA FC,30,30,30,30,30,30,00
85 DATA C8,C8,C8,C8,C8,C8,70,00
86 DATA C8,C8,C8,C8,C8,D0,E0,00
87 DATA C4,C4,C4,D4,D4,EC,C4,00
88 DATA C8,C8,70,20,70,C8,C8,00
89 DATA C8,C8,C8,70,20,20,20,00
90 DATA F8,08,18,30,60,C0,F8,00
91 DATA F0,C0,C0,C0,C0,C0,F0,00
92 DATA 00,80,C0,60,30,18,08,00
93 DATA F0,30,30,30,30,30,F0,00
94 DATA 20,70,D8,00,00,00,00,00
95 DATA 00,00,00,00,00,00,F8,F8
96 DATA 40,60,30,10,00,00,00,00
97 DATA 00,00,70,18,78,98,78,00
98 DATA C0,C0,F0,C8,C8,C8,F0,00
99 DATA 00,00,70,C8,C0,C8,70,00
100 DATA 18.18.78,98,98.98,78,00
```

```
101 DATA 00,00,70,C8,F8,C0,70,00
102 DATA 30,68,60,F8,60,60,60,00
103 DATA 00,00,78,98,98,78,18,70
104 DATA C0,C0,F0,C8,C8,C8,C8,00
105 DATA 60.00,60,60,60,60,F0,00
106 DATA 30,00,30,30,30,30,B0,60
107 DATA 60,60,68,70,60,70,68,00
108 DATA 70,30,30,30,30,30,78,00
109 DATA 00,00,E8,D4,D4,D4,D4,00
110 DATA 00,00,F0,C8,C8,C8,C8,00
111 DATA 00,00,70,C8,C8,C8,70,00
112 DATA 00,00,F0,C8,C8,F0,C0,C0
113 DATA 00,00,78,98,98,78,18,18
114 DATA 00,00,F0,C8,C0,C0,C0,00
115 DATA 00,00,78,C0,70,18,F0.00
116 DATA 60,60,F0,60,60,68,30,00
117 DATA 00,00,D0,D0,D0,D0,68,00
118 DATA 00,00,C8,C8,C8,50,20,00
119 DATA 00,00,C4,D4,D4,D4,68,00
120 DATA 00,00,C8,70,30,58,88,00
121 DATA 00,00,98,98,98,78,18,70
122 DATA 00,00,F8,18,30,60,F8,00
197 '
198 ' Caricatore
199 '
200 COLOR 15,4,4
210 SCREEN 0
220 WIDTH 40
230 KEY OFF
240 DEFINT A-Z
250 LOCATE 14,7
260 PRINT"WAIT PLEASE"
270 B2=BASE(2)+31*8
280 FOR C=1 TO 91
290 FOR D=0 TO 7
300  READ A$
310  AD=B2+C*8+D
320  DA=VAL("&H"+A$)
330  VPOKE AD,DA
340 NEXT
350 NEXT
```

COLLEGE

Promuove il tuo lavoro con 110 e lode

COD. CL 219 * Disk 5,25 DS.DD 48 TPI	L. 20.000	conf. da 10 pz.
COD. CL 226 * Disk 5,25 DS.DD 96 TPI	L. 22.000	conf. da 10 pz.
COD. CL 227 * Disk 5,25 DS.4D 1,2 MEG	L. 35.000	conf. da 10 pz.
COD. CL 228 * Disk 3,50 DS.DD.	L. 25.000	conf. da 10 pz.
COD. BK 200 * Disk 5,25 DS.DD. Bulk cad.	L. 1.100	conf. da 100 pz.
COD. BK 229 * Disk 5,25 96 TPI Bulk cad.	L. 1.400	conf. da 100 pz.
COD. BK 224 * Disk 5,25 DS.4D. 1,2 MEG Bulk cad.	L. 2.600	conf. da 100 pz.
COD. BK 205 * Disk 3,50 DS.DD. Bulk	L. 2.600	conf. da 100 pz.
COD. DP 204 * Duplicazione floppy 5,25 formato Pc/Ibm	L. 650	pz. 500 minimo
COD. DP 202 * Duplicazione floppy 5,25 formato Commodore 1 lato	L. 380	pz. 500 minimo
COD. CT 220 * Cartelletta proteggi software	L. 1.800	
COD. PD 211 * Kit pulizia drive 5,25	L. 20.000	
COD. PD 231 * Kit pulizia drive 3,50	L. 25.000	

Prezzi comprensivi di IVA

BUONO di ORDINAZIONE

Nome .....
Cognome .....
Indirizzo ..... N. .....
Cap ..... Città .....
Prov. ..... P. IVA e/o Cod. Fisc .....

Vogliate inviarmi in contrassegno

QT ..... Cod. art. ..... L. .....
QT ..... Cod. art. ..... L. .....
QT ..... Cod. art. ..... L. .....
QT ..... Cod. art. ..... L. .....

Paghero al postino più spese postali
Per ordini telefonici: 02/5471321 - 5472470

Ordini da inviare a: PHONO-PLAST s.r.l. - Via A. Grandi, 50 20068 Peschiera Borromeo (Milano) Telex 326498 Phono-I

LA PERFEZIONE DIVENTA MITO

MITO - 5 1/4" Floppy 48 TPI
Doppia Faccia - Doppia Densità
Garantito al 100% - Velocità di
registrazione 5800 BPI
600.000 bytes unformatted.

le misure della perfezione

RECOVERY SERVICE - Un nostro servizio esclusivo. Cosa è il Recovery Service? E uno scudo a protezione del vostro lavoro. Se per un incidente qualsiasi: macchie di caffè, di cioccolato o impronte, il vostro disk dovesse danneggiarsi, la MICROFORUM è in grado di recuperare i dati senza alcun esborso da parte vostra.

Microforum
MANUFACTURING INC.
TORONTO - CANADA

La *MICROFORUM MANUFACTURING INC.* è interessata all'ampliamento della propria rete distributiva. Per qualsiasi contatto scrivere anche in italiano.

944/A St. Claire Ave. West TORONTO, CANADA M6C 1C8 Tel. (416) 656-6406 - Tlx. 06-23303 MICROFORUM TOR Telefax (416) 656-6368

STUDIO C.SANTORI Roma

QL BULLETTIN BOARD

a cura di Eros Forenzi

Il Sinclair QL è fuori produzione ormai da più di un anno e mezzo, eppure non è ancora morto. Continua a vivere non certo grazie all'incoraggiamento o all'apporto della ex Sinclair (ora Cambridge Computing) nè grazie ad Alan Sugar, che, dopo aver acquistato per un boccone di pane l'attività di sir Clive ha deciso di buttare il QL nella spazzatura e di concentrarsi nella produzione e nel lancio di nuovi computer Amstrad (per il momento poco più che buchi nell'acqua).

Il QL vive ancora grazie alla fiducia di numerose software house inglesi, tedesche, francesi, americane, e all'apporto ed entusiasmo di numerosissimi possessori di questo computer.

Ma quanti sono i QL in giro per il mondo?

Beh, circa 80000 si trovano in Inghilterra, poi ce ne sono 15000 nella Germania Occidentale, 20000 in Italia, 10000 in Francia e almeno 50000 sparsi per il resto dell'Europa e del mondo .

Ce ne sono un po' dappertutto, negli U.S.A., in Nuova Zelanda, in Australia ma anche in posti inimmaginabili, come nello Sri Lanka o in Arabia Saudita e così via.

Credo che 170000 Ql siano una bella cifra; certo, siamo lontani dai milioni di Spectrum o Commodore 64 venduti fin'ora, ma non dobbiamo dimenticare che il QL-ista tipo non è il ragazzino di dieci anni che passa il tempo a giochicchiare, bensì è un hobbista appassionato della macchina che possiede, che ama programmare, e che vorrebbe farsi al più presto un sistema completo (unità centrale, memoria aggiuntiva, disk drive e stampante).

Il Ql è poi un computer dal quale sanno tirar fuori bellissimi programmi non solo gli inglesi (che ad esempio quasi monopolizzano la produzione di software per lo Spectrum), ma anche gli italiani, i francesi, i tedeschi ecc. (Programmare è qui da intendersi nel senso di capacità di scrivere routines o programmi complessi in linguaggio macchina).

Morale? Pochi utenti appassionati e capaci possono tenere in vita e far prosperare macchine date più volte per morte! Caro Alan Sugar, impara l'Italiano e leggi anche tu.

Noi di Personal Computer siamo consci di tutto ciò, ed è per questo che desideriamo dare una mano ai 20000 QListi italiani, un aiuto che si concreterà nel mettervi al corrente di tutte le le novità o notizie che riguardano il vostro computer.

Informazioni magari semplici semplici e risapute per qualcuno, ma che possono interessare molti altri nuovi arrivati.

Iniziamo con la prima raffica di utili notiziole. (Hints & Tips)

• Quante sono le versioni della Rom del QL?

Torneremo presto sull'argomento con un articolo più approfondito che tratterà le rom e.....i bug.

Per ora ci limitiamo a dire che le versioni della Rom sono almeno sette: FB, PM, AH, JM, JS, JSU e MG. Quest'ultima è stata poi prodotta in alcune sottoversioni personalizzate per alcuni paesi europei.

Abbiamo così la Rom italiana MGI, la francese MGF, la spagnola MGE, la tedesca MGG, e così via, per finire con la Rom greca.

• Quante sono le versioni del QDOS?

Anche qui approfondiremo l'argomento in un prossimo articolo.

Per ora eccovi l'elenco delle diverse versioni del QDOS: 0.08, 1.00, 1.01, 1.02, 1.03, 1.10 e 1.13.

• Avete un QL JM e vorreste un reset come quello che si ottiene sullo Spectrum con RANDOMIZE USR 0?

Digitate in modo diretto o da programma il comando CALL 360 seguito dal tasto ENTER.

Questo reset funziona nel 90% dei casi; alcune volte infatti il computer si blocca o addirittura inizia a fare cose strane (in questo ormai lo sappiamo non è secondo a nessuno!!).

Uno dei casi in cui Call 360 non funziona si ha quando si è in modo monitor e sono stati impartiti comandi del tipo PRINT DATE$.

• Le cartucce microdrive non hanno tutte la stessa capacità in K.

Questo come sapete dipende da molti fattori; esistono però due rimedi efficaci.

E' possibile ad esempio ottenere dai due ai dieci K in più se si tiene premuta con un dito la cartuccia per l'intera durata della formattazione. La pressione da esercitare non deve essere nè troppo forte nè troppo debole.

Il miglior consiglio è quello di provare varie volte con pressioni diverse fino a trovare quella che dà il miglior risultato (non tutte le aperture dei microdrive sono infatti uguali). In un 5% dei casi la cartuccia così formattata presenta alcuni settori difettosi (potete verificarlo con con la Cartridge Doctor o con il Qdoctor). Potete rimediare a questo inconveniente riformattando la cartuccia.

Il secondo consiglio lo avrete prossimamente in un atricolo più dettagliato.

A proposito, il record (nel senso "ginnico", non informatico) di formattazione del mio QL JM è di 238/239 settori, mentre un tedesco ha dichiarato di aver raggiunto i 251 settori. Inoltre questo trucchetto funziona anche con i microdrives dello Spectrum.

Ah, ricordo che il QDOS prevede la formattazione al massimo di 256 settori, per cui .....non premete troppo.

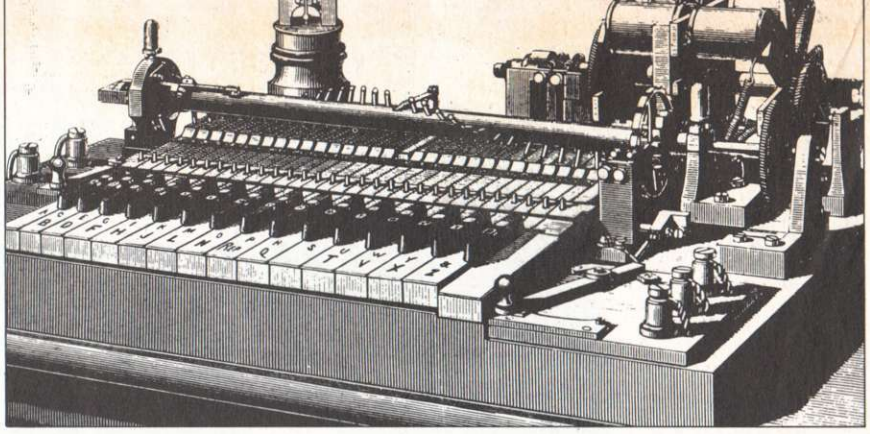
• Differenze hardware:

Oltre alle diverse versioni Rom e Qdos esistono anche altri due indicatori che differenziano un QL da un'altro.

Il primo è la release hardware, che corrisponde più o meno alle issue 1, 2, 3, 4 e 6 dello Spectrum.

Si tratta di modifiche più o meno notevoli della parte hardware del computer.

Questo indicatore si trova inciso sul contenitore del QL, proprio sotto i tasti



funzione ed è rappresentato dalla lettera "D" seguita da un numero. A fianco si trova invece il numero di matricola del computer. Il mio QL ad esempio è un JM con QDOS 1.03, release hardware D09 e matricola 30709. L'evoluzione delle modifiche ha raggiunto almeno la D17, per cui il mio QL è un po' vecchiotto. In un negozio di MIlano ho poi visto in vendita un QL MGI con D14 e matricola 84000 (quindi anche noi italiani non siamo poi così maltrattati!).

Prima di acquistare ricordatevi di voltare il computer e guardatene la matricola; non tutti i QL che ancora ci sono nei negozi sono nuovi. Molti sono dei vecchi esemplari delle prime serie prodotte, ritirati dalla Sinclair perchè difettosi, rabberciati alla meno peggio e quindi rimessi sul mercato.

Vi chiederete quali siano le differenze tra le varie versioni;
beh, non se ne conoscono molte, comunque ce n'è senz'altro più d'una (a che punto arriva l'ovvietà umana!). In un articolo pubblicato sulla rivista inglese Your Computer si diceva ad esempio che si era dovuta attendere la modifica D12 prima di poter vedere un QL perfettamente funzionante.

C'è poi un altro difettuccio ben più grave: la maggior parte dei QL inferiori a D14 hanno le porte network che funzionano male o che non funzionano del tutto, e in questi casi non c'è una nuova Rom o un Toolkit 2 che possano riparare al guaio.

I bug hardware sono i più malefici, difficili da eliminare se non ..... cambiando il computer.

Un'altra differenza sta nel minore surriscaldamento dei QL più recenti (se il computer "scalda" troppo può andare in crash), poi c'è la porta per le cartucce Rom che non funziona su molti QL (ossia non tiene dentro ben salda la rom esterna) o che oppure non riconosce le cartucce Rom inserite (difetto di alcuni QL MGI).

L'altro indicatore è l'hardware revision level, che si trova sulla scheda principale del computer (dovete quindi aprirlo).

E' rappresentato dalla lettera "A" seguita da un numero: A5, A4 ecc.. Anche in questo caso non si conoscono bene quali differenze ci siano tra i vari revision levels, ma ad esempio, sempre per ritornare al difetto delle porte network, Tony Tebby nel manuale del Toolkit 2 afferma che non funzionano sulla maggior parte dei QL con serie inferiori a D14 .... a meno che non abbiano un revision level A5, come in effetti è per molti QL destinati al mercato europeo.

• Toolkit & company:

Tony Tebby è veramente un gran lavoratore; ormai le versioni del suo toolkit si moltiplicano come conigli.

In principio fu il QL Toolkit v1.0, venduto dalla Sinclair su cartuccia microdrive.

Poi vennero le versioni 1.16 e 1.17 adattate per le interfacce disco. In seguito Tebby scrisse il famoso Toolkit 2 su rom, seguito quasi subito dalla versione 2.02, con in più il comando Print-Using.

Arriviamo così oggi alle versioni 2.08 e 2.10 inserite sulle ultime Superqboard della Sandy, per finire con la 2.11 che si trova sulla nuovissima Trump Card della Miracle.

Questo Toolkit 2 su rom abbinato ad un QL JS con almeno D14 rappresenta forse il massimo in fatto di comandi aggiuntivi e correzione di bug. C'è da dire che, pur essendo su rom, il Toolkit 2 occupa un paio di K di ram, inoltre rallenta la velocità del computer di un 4%.

• Chissà quanti sono i programmi commerciali disponibili sul mercato per il nostro Sinclair...

Dovrebbero essere almeno 300, senza tener conto naturalmente dei giochini stupidi in superbasic o dei programmi composti da una routine di qualche centinaio di bytes apparsi nei primi mesi di vita del QL. Se a questi aggiungiamo tutti i programmi o listati più o meno lunghi apparsi sulle riviste specializzate supereremmo di molto quota mille.

Certo, i programmi commerciali di rigore (ossia veramente belli e potenti) non sono più di un centinaio, nulla di paragonabile ai 6000 e oltre programmi disponibili per lo Spectrum.

Comunque non c'è da lamentarsi troppo, le software house continuano a sfornare ottimi prodotti al ritmo di 4 o 5 al mese, di qualità a volte superba, in grado di sbeffeggiare prodotti analoghi per Amiga o Atari ST.

Una caratteristica del mercato del software QL è poi quella di avere differenti versioni di uno stesso prodotto.

Molte volte ci sono programmi molto diversi tra loro come potenza e versatilità, ma distinti solo da un numerino che ne identifica la versione. E' questo il caso di Super Sprite Generator della Digital Precision, uscito nelle versioni (quelle che mi ricordo) 1.0, 2.0, 3.5 e 4.0. Potremmo poi ricordare il compilatore Qliberator, le cui versioni principali sono la 1.0, la 2.0, la 3.0 e la 3.1, oppure il bellissimo front end della Talent: QIMP, uscito nelle versioni 1.0 e 1.50.

Il profano molte volte non fa caso a questi numerini...eppure ...

Lo Sprite Generator 4.0 ha ad esempio le keywords modificate per ottenere i massimi risultati in programmi compilati con TURBO.

Le versioni dalla 1.14 in poi di Turbo sono invece compatibili con Taskmaster e con il front end QRAM, e così via.

Anche i 4 programmi PSION, Quill, Archive, Abacus ed Easel hanno le loro brave versioni.

Alcune volte si tratta di programmi tradotti nelle diverse lingue nazionali (come la 2.23 per l'Italia), altre volte si tratta di tentativi di eliminare i numerosi bug purtroppo presenti in questi pur validi programmi.

Ricorderete con odio le iniziali versioni 1.0, 1.01 e 1.03; le cose migliorarono con la 2.0 ... ma non troppo.

Molto meglio la versione 2.3, preceduta almeno dalla 2.21, 2.23 e 2.25.

L'ultimo sforzo della Psion è stata la versione 2.35 di tutti e quattro i programmi e la 2.36 del solo Archive.

Ho poi notato su una rivista tedesca uno strano riferimento ad una versione 2.45 di Archive ... sarà vero?
Ultimo consiglio: se avete un QL 640K e volete far girare in multitasking i 4 Psion, procuratevi la versione 2.35.

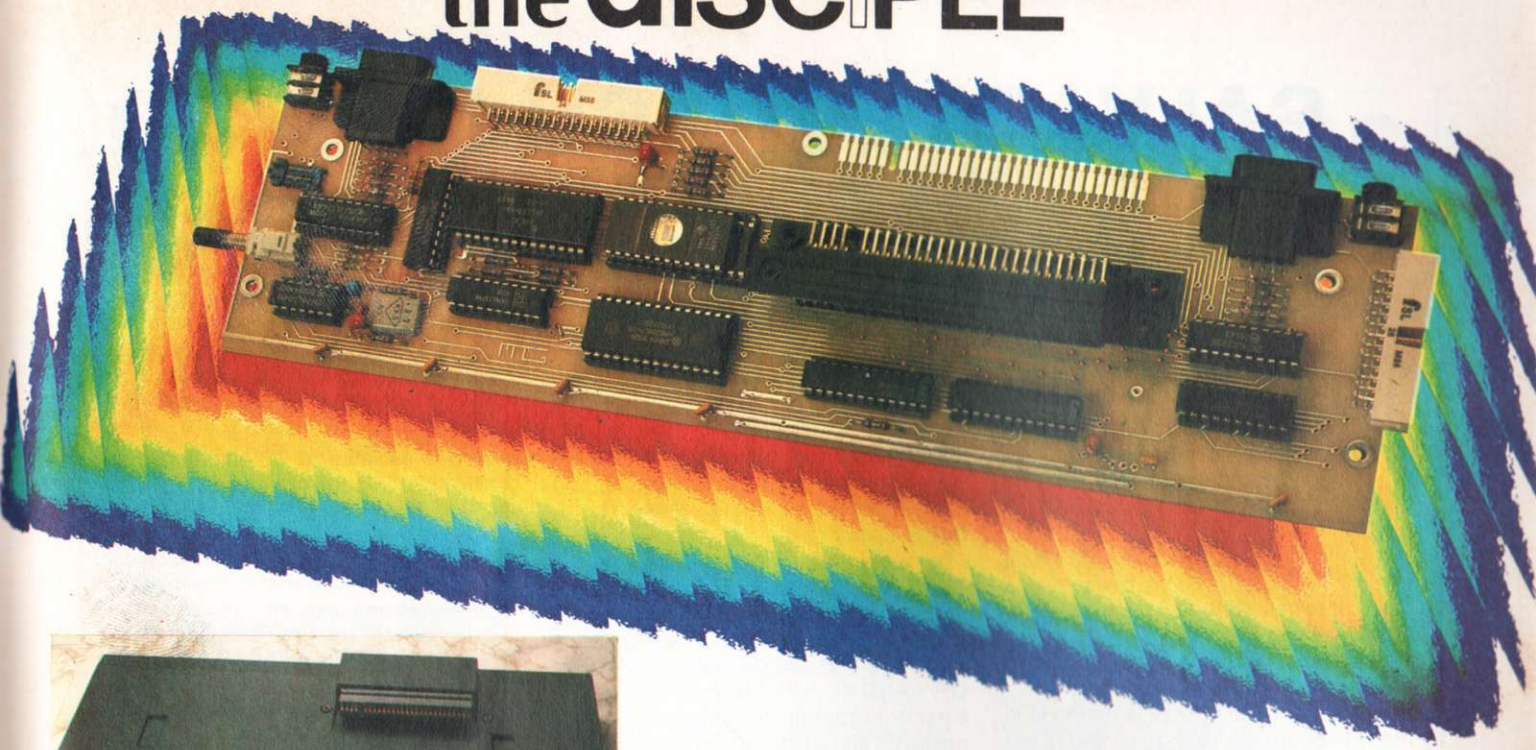
Pare infatti che ci siano state positive modifiche alla gestione della memoria, in modo da ottenere migliori risultati nell'uso con programmi di gestione del multitask come QSWITCH o Taskmaster.

E per questo numero penso possa bastare.

Nei prossimi mesi parleremo di moltissime cose: espansioni Ram, novità software, comunicazione tra QL e Spectrum via RS 232 e dalle porte Network (sì, anche dalla network), e così via.

Arrivederci a presto.

the DISCIPLE



LA PIÙ POTENTE SCHEDA MULTIFUNZIONE MAI PRESENTATA PER LO SPECTRUM 48/128 K:

* **INTERFACCIA DISCO:** consente di collegare fino a due dischi drive da 3,5" a 5 1/4", formatta 800 Kb per disco; carica un programma da 48 Kb in 3,5 secondi; utilizza due sintassi; una compatibile con i comandi dell'interfaccia 1 ed una semplificata.
* **TASTO SNAPSHOT:** trasferisce su disco qualsiasi programma in memoria semplicemente premendo l'apposito tasto.
* **INTERFACCIA STAMPANTE:** compatibile con qualsiasi stampante parallela; utilizza le istruzioni Basic LLIST, LPRINT, COPY consentendo l'hard copy del video senza software ed hardware ausiliario.
* **DUE INTERFACCE JOYSTICK** compatibili con lo standard Sinclair e Kempston.
* **NETWORK:** consente di collegare in rete fino a 64 Spectrum che possono sfruttare un solo drive ed una sola stampante.
* **COMPATIBILITÀ:** compatibile con qualsiasi Spectrum, anche con il nuovo 128+2, sia in modo 48 che in modo 128.

DISCIPLE **L. 199.000**
DISCIPLE Jr. **L. 149.000**
3.5" - alimentatore incorporato - 800 Kb formattati

INOLTRE DISPONIBILI

MUSIC MACHINE **L. 129.000**
Batteria polifonica digitale ad 8 voci, campionatore (digitalizza qualsiasi suono proveniente dall'esterno, anche la voce, e permette di modificarlo tramite lo studio dell'onda), synth, eco, riverbero, 3 porte MIDI (in, out, through). Tutto questo sul Vs. Spectrum: ascoltare per credere! La Music Machine viene fornita completa di Microfono.

RAM PRINT **L. 99.000**
Interfaccia stampante per Spectrum, completa di cavo per collegamento ad una qualsiasi stampante parallela e porta joystick. Inoltre sulla ROM dell'interfaccia è presente un potente programma di Word Processing e tutto il software di gestione della stampante, entrambe richiamabili da basic.

QL SUPERDISK INTERFACE **L. 149.000**
Interfaccia disco e stampante parallela con toolkit 2 residente in ROM.

QUINTERAM **L. 199.000**
Espansione a 640 Kb per sinclair QL. Si monta internamente senza saldature. Inoltre una ROM contiene svariati comandi aggiuntivi utilissimi per ramdisk, multitasking ecc. ecc.

DISK DRIVE 3.5" **L. 250.000**
DOPPIO DISK DRIVE 3.5" **L. 399.000**
800 Kb formattati. Completi di alimentatore incorporato e cavo di collegamento alle relative interfacce per Spectrum, QL o MSX.

DISCHETTI 3.5" **L. 35.000**
Box da 10 dischi Nashua doppia faccia/doppia densità/135 TPI.

MEGABYTE

VENDITA ANCHE PER CORRISPONDENZA

Piazza Duomo 17 - 25015 DESENZANO DEL GARDA - BRESCIA Telefono 030/9144880
PEIS MAILBOX CA 0355 - Telex 520560 INTSVI (Destinatario 0355)

SALVATAGGIO SU NASTRO DI FINESTRE VIDEO

di Ezio Boscani

Lo schermo dello Spectrum non si applica a certi trucchi come potrebbe essere il salvataggio od il caricamento di finestre video.

La semplicità con cui però è stata scritta al routine di LOAD in ROM, permette una facile manipolazione della stessa, fino a raggiungere diversi risultati, quali il coloramento diverso dal solito sul bordo che troviamo in diversi giochi.

Il fatto che la routine tratti la memoria in modo sequenziale, usando un INC IX per il caricamento (ma ciò vale anche per il salvataggio), ci dà la possibilità, rilocandola opportunamente, di farla funzionare in modo diverso.

Chi andrà a disassemblare la routine, troverà che è stato posto un POP IX al suo posto: ciò serve per prelevare da uno stack che è stato opportunamente creato, l'indirizzo ove caricare il successivo byte proveniente dalla cassetta.

Veniamo ai due programmi: il primo è il caricatore in L/M per le due routine di LOADWIN$ e SAVEWIN$, che stanno entrambe in un totale di 1K.

Il secondo è una utility per salvare le vostre finestre caricandole da nastro/mdrive e salvarle poi su nastro.

Quando lo avrete digitato, salvatelo con autostart alla linea 9000, dando GOTO 9900; al messaggio di STOP, se vorrete salvare anche la routine, date CONT.

Il programma si presenta con un MENU' di 4 opzioni: la seconda vi mostra la schermata da elaborare con i seguenti comandi:

- Tasti cursore : vi permettono di spostare il cursore su tutto il video;
- Tasto 0 : setta l'origine della finestra da salvare, che sarà ricordata da un altro cursore fisso;
- Tasto EDIT (CS+1) : permette di passare dal video con gli attributi ad un video in carta nera e inchiostro bianco e viceversa: ciò quando volete elaborare schermate che in certi punti hanno carta e inchiostro uguali e fanno sparire il cursore di editing;
- Tasto ENTER : ritorna al menù principale e, se l'origine è stata settata, verranno mostrati i parametri di salvataggio.

Se per caso avete sbagliato a settare l'origine, tornate al menù principale con ENTER, quindi rientrate a settare le coordinate della finestra da salvare.

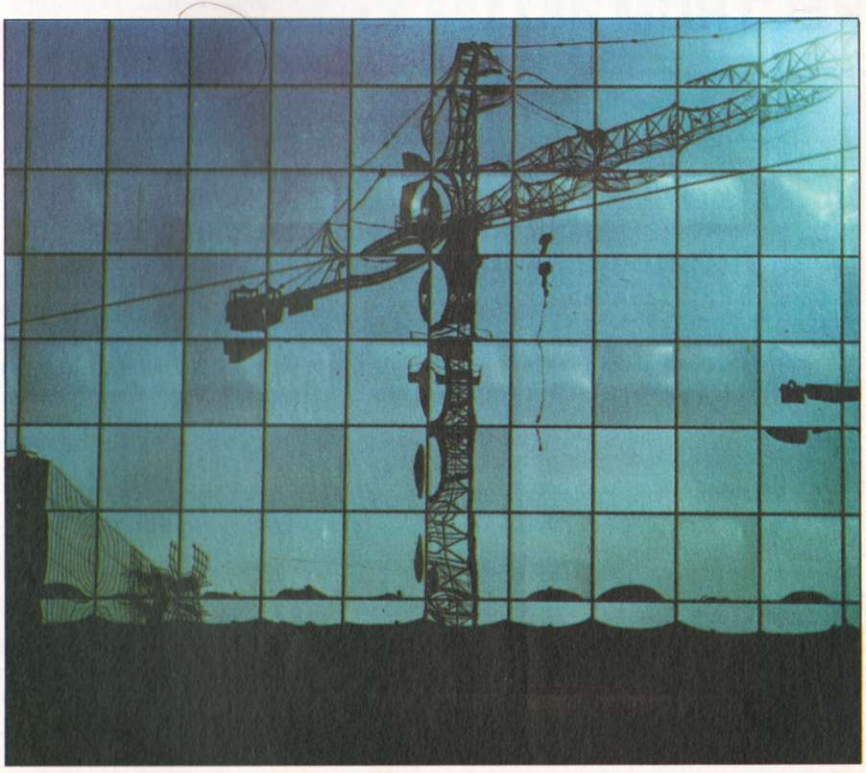
Per far agire la routine di LOAD, basta dare RAND USR 64500: la routine di save, invece, deve essere innescata con un comando che può sembrare complesso, ma che è abbastanza lineare: RAND X OR Y AND XL + YL * USR 65000, dove x e y sono le coordinate dell'origine della finestra (in alto a sinistra) come nell'AT del print; XL e YL sono la larghezza e l'altezza della finestra sempre in caratteri.

Nota: La routine di SAVE, crea un HEADER di 4 bytes che contiene le informazioni che vengono passate da BASIC e che servirà poi alla routine di LOAD per creare lo stack di lavoro.

Per evitare che i dati dello stack vadano ad influire sulla RAMTOP, consiglio di abbassare la stessa al valore di 50499, in quanto il salvataggio di uno schermo completo, comporta la creazione di uno stack di lavoro lungo 13824 bytes che partono in "discesa" dall'indirizzo 64400.

La routine non è rilocabile se non a mezzo di un assemblatore ed è protetta da crash che possono succedere in caso di pressione del tasto BREAK o da Tape Loading Error col rientro al BASIC con un messaggio 0 OK : cioè il programma continua nonostante l'errore.

Le due routine sono contigue in memoria, comunque per chi volesse salvarle separatamente, la LOADWIN$ sta a 64500 ed è lunga 500 bytes, la SAVEWIN$ sta a 65000 ed è lunga anch'essa 500 bytes. Attenzione che la SAVEWIN$ occupa parte dello spazio degli UDG.



HACKERS & CRACKERS

a cura di Fabio Berno

Non saranno in molti a non saper che "hacking" significa, in pratica, alterare un programma in modo che lavori diversa mente da come il suo programmatore ha inteso. Scendendo sul concreto, è possibile, per i più esperti, modificare un gioco facendo sì che le vite a disposizione non diminuiscono quando si muore, permettendo giocate infinite. E' anche possibile aggirare le protezioni anti-copia dei programmi protetti: ma, avvertiamo, questa attività, oltre che illegale, è talmente dispendiosa in termini di tempo che non ne vale molto la pena (per copiare i programmi ci sono altri modi più veloci). Lo scopo di questo articolo è solo quello di farvi divertire di più attraverso i giochi, una volta che avrete raggiunto lo stadio di abilità che è necessario.

Gli strumenti essenziali per qualsiasi hacker sono essenzialmente due: un disassemblatore e la pazienza. Per il primo, posso consigliarvi DevPac della Hisoft, per la seconda pensateci voi. Cosa serve sapere ? Una sola cosa: il linguaggio macchina. Essendo l'articolo rivolto allo Spectrum, è naturale che il L.M. da conoscere è quello dello Z80; coloro che non lo conoscono, però, non voltino pagina: mica è così difficile. Provino a seguire lo stesso il discorso e chissà.. . Comunque, posso consigliare alcuni libri: primo tra tutti, il libro di R.Zaks "Programmazione dello Z80 (disponibile anche in italiano, Jackson ed.); poi, in inglese, "Spectrum Machine language for the Absolute Beginner" della Melbourne House. Il libro di Zaks è caldamente consigliato anche ai più esperti.

Dopo questa premessa, si può iniziare: cosa servirà ora ? Naturalmente, un gioco difficile da fare, dove proprio non riuscite a combinare granchè. I puristi dello sport alla de Coubertin non storcano troppo il naso: ci sono milioni di milioni di frustrati dei videogames (come il sottoscritto) che ambiscono almeno di capire lo spirito di un gioco, senza essere buttati subito fuori. La cosa più semplice da fare è ovviamente, quella di usare un gioco non protetto, ed è l'assunzione che faremo qui; per le sprotezioni, leggetevi il "serial" in 8 puntate da noi pubblicato sui numeri 5/6/7/8/9/10/11/12. Come distinguere se un programma è protetto o no ? Il modo più semplice è confrontare la modalità di caricamento (righe gialle e blu, tono dell'header..) con quella dei programmi normali; questa procedura funziona nella maggior parte dei casi. Un esempio ? Il famosissimo Jet Set Willy.

Ora, scelto il gioco, che si fa ? Si carica il disassemblatore (o monitor) nello Spectrum, ad una posizione da scegliere con cura e comunque in memoria "alta", diciamo sopra l'indirizzo 55000, per non interferire con il codice del gioco. I più esperti potranno leggere le informazioni necessarie al posizionamento del monitor nell'header del gioco, e piazzare il monitor stesso dove non è presente nulla di vitale al gioco stesso, e cioè dopo.

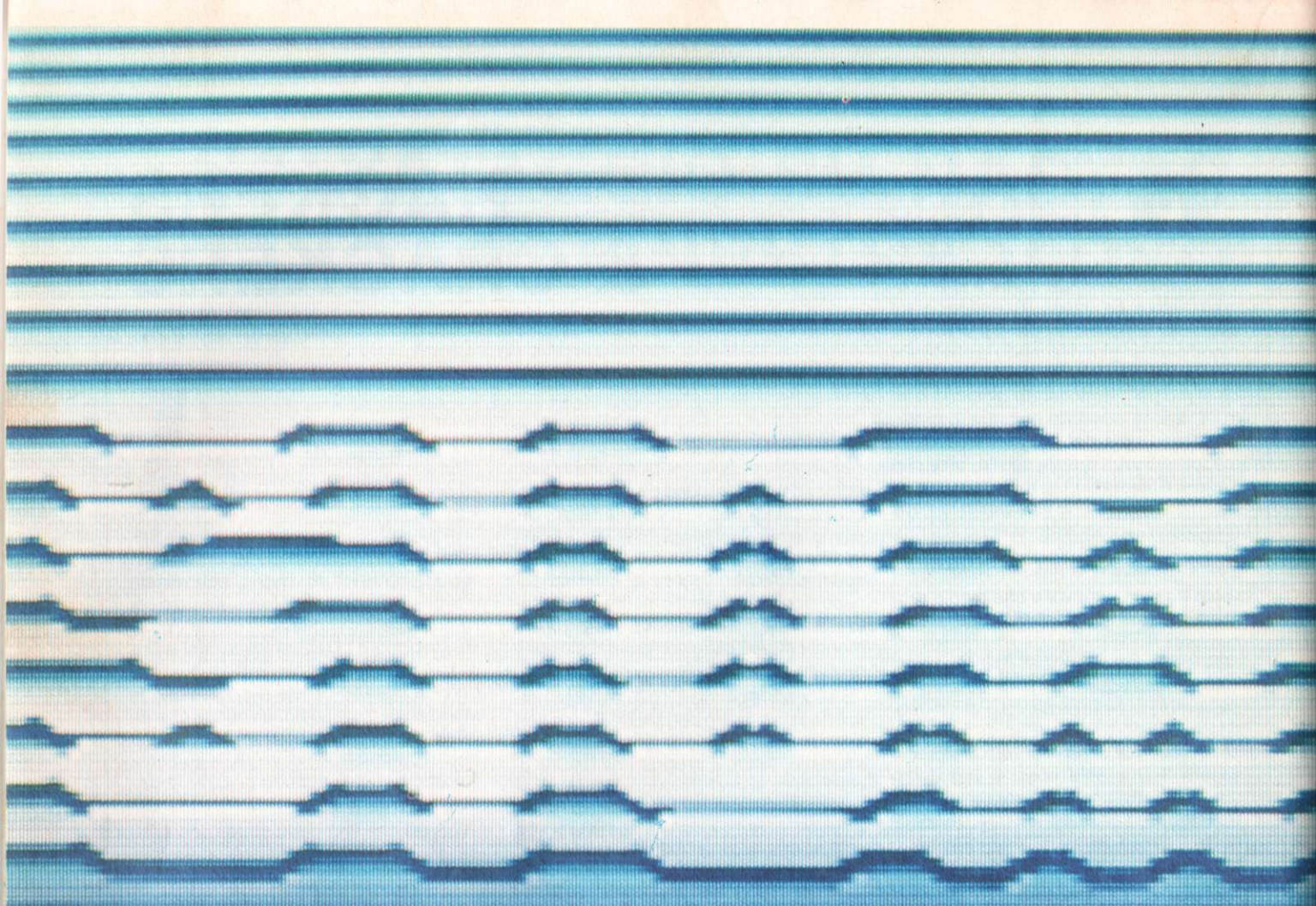
Bisogna ora caricare il gioco stesso nello Spectrum: fatelo con MERGE, non con LOAD; potrete così, al termine del caricamento, listare il programma BASIC di controllo, e scoprire alcune cose interessanti quali l'indirizzo della RAMTOP (che è l'indirizzo di CLEAR + 1; la RAMTOP è l'indirizzo massimo utilizzabile dal BASIC) e l'indirizzo di partenza del programma in L.M., di solito indicato dalla istruzione USR: una istruzione RANDOMIZE USR xxx indica che il programma il L.M. parte all'indirizzo "xxx".

Andiamo avanti. Supponiamo che il programma in L.M. del gioco sia in memoria, e che abbiate già fatto partire l'esecuzione del monitor. Cosa si deve cercare? Dunque, nonostante la grande varietà di giochi, le istruzioni assembler che implementano il gioco non sono poi moltissime: di solito, sempre riferendosi al numero di vite a disposizione, si deve cercare qualcosa come LD A,x e come LD (nn),x , dove "x" è il numero di vite con cui si parte e "nn" l'indirizzo dove le vite sono immagazzinate (chiamato anche "paradiso"). In seguito si dovrà cercare qualcosa tipo LD A,(nn):DEC A (il duepunti significa "in sequenza), oppure LD HL,(nn):DEC HL , che indicano dove le vite sono decrementate. Vedrete che ci saranno pochi indirizzi che hanno tali istruzioni in sequenza, e potrete quindi verificarli abbastanza agevolmente ed eventualmente modificarli con dei NOP tramite delle POKE di valori 0 date dall'interno del programma BASIC, POKEs da fare all'indirizzo dove sono presenti i DEC naturalmente. Per quanto riguarda quei giochi dove è importante anche il tempo a disposizione, bisogna cercare contatori a due bytes e il loro decrementi, come LD HL,xx:LD (nn),HL e LD HL,(nn):DEC HL.

Un piccolo trucco: cercate, sempre tramite il monitor, il testo del messaggio "GAME OVER" (o quello equivalente che tante volte vi ha frustrato), cercate poi l'indirizzo a cui si riferisce (cercando LD HL,nn oppure LD DE,nn dove "nn" è l'indirizzo del messaggio maledetto), poi cercate cosa fa saltare al codice che fa stampare il messaggio, e vedrete che da qualche parte lì intorno ci sarà l'istruzione che decrementa le vostre vite e aumenta il vostro nervoso quando giocate: stroncatela senza pietà.

Vedrete che fino qui la strada non è poi difficile. Con un po' di pratica, sarà ancora più semplice. Le difficoltà maggiori sono ovviamente nella sprotezione dei programmi, senza la quale non potrete fermarli e pasticciarvi sopra. Infatti, molti giochi incorporano routines in L.M. nel caricatore BASIC in modo che siano impossibili da MERGEare: questo L.M. contiene di solito un altro caricatore che carica il gioco in memoria e lo esegue, impedendo ai curiosi o ai pirati (i secondi sono i più numerosi) di accedervi. In questi casi, di solito si crea un falso header al programma, registrandolo sopra l'originale che prima va però esaminato (ci sembrava ovvio, ma lo diciamo lo stesso). Un creatore di header è stato da noi pubblicato sul numero 2 della rivista.

Vogliamo terminare con un piccolo esempio di "hacking" su uno dei più diffusi giochi per Spectrum: si tratta del famoso Bomb Jack 2 della Elite. Esso ha un sistema di protezione veramente.. inesistente, perchè il BASIC può essere MERGEato e LISTato senza difficoltà; il codice macchina viene caricato separatamente e lanciato con un semplice RANDOMIZE USR xxx, consentendo così di vedere dove parte. Il L.M., inoltre, non è cifrato (talvolta capita anche questo), e così può essere guardato nel modo più semplice con il solito disassemblatore/monitor. Il listato BASIC da sostituire a quello originale è il seguente, e contiene anche le POKE necessarie per avere infinite vite.



SIOA

Salone dell'Informatica della Telematica e della Organizzazione Aziendale

9-13 Aprile 1988
Fiera di Bologna

Promozione:
ANIE Associazione Nazionale Industrie Elettrotecniche ed Elettroniche

Ente Autonomo per le Fiere di Bologna

Fondazione Guglielmo Marconi

Organizzazione:
Consorzio SIOA
Via Napoli, 20 - 40139 Bologna
Tel. 051/452936-466911
Telex 583131 EXPOIT I

UN CONVERTITORE ANALOGICO DIGITALE PER LO SPECTRUM

di Mario Mella

Sempre più negli ultimi anni si sente parlare di informatizzazione della casa, di computer che tengono sotto controllo il riscaldamento, o verificano con una serie di sensori che non vi siano presenze anomale nella casa, assolvendo così anche il compito di antifurto. Un altro possibile utilizzo è quello di accendere le luci di casa quando si verificano contemporaneamente le condizioni che la luminosità naturale sia scesa sotto una certa soglia e che qualcuno sia in casa. Questo per non parlare dei possibili campi di applicazione hobbistici che il computer può asservire. Tanto per fare un esempio, se qualcuno ha l' hobby della fotografia con stampa domestica delle immagini, il computer, oltre a controllare la temperatura dei bagni di sviluppo, può calcolare l' esatta esposizione e, dopo aver stabilito il numero di secondi, pilotare l' ingranditore conseguentemente.

L'elenco di possibili applicazioni è illimitato e, per citare una frase ormai un po' abusata, l'unico limite è dato dalla fantasia dell' utente. Tuttavia, il tipico possessore di uno Spectrum, a questo punto si chiederà: "Ma come faccio a misurare una temperatura o una intensità luminosa con il mio computer?". Infatti L'unica interfaccia di input che lo Spectrum ha verso il mondo, è la tastiera la quale può ricevere solo dei "sì" o dei "no" per ogni tasto, corrispondenti alla pressione o meno del dito su quel tasto. Si può anche parlare dei joystick, ma anche qui, l'input è di tipo binario: on-off, in quanto le manovre sul joystick pilotano dei microinterruttori. Per tutte le misurazioni di tipo continuo lo Spectrum, non è in grado di dirci nulla.

E' tuttavia possibile realizzare con poca spesa un interfaccia che ci permetta di far leggere allo Spectrum una tensione continua compresa tra 0 e 5 volt, che viene trasformata in una parola binaria facilmente leggibile dal nostro amato computer. Si tratta di un convertitore analogico/digitale (ADC = Analog to Digital Converter). Questo componente sta alla base di tutti i sistemi di controllo di processo utilizzati nelle fabbriche automatizzate di cui si sente tanto parlare. Infatti è vero che i robot di montaggio sono pilotati da computer, ma questi computer, prima di impartire ai robot le operazioni da fare devono verificare se il pezzo sul quale operare è presente, è nella corretta posizione, se la temperatura di un forno è corretta o la pressione di una certa pompa non è troppo bassa. Per acquisire tutti questi dati si fa uso di convertitori AD.

Una volta acquisite le informazioni, il computer deve impartire degli ordini, azionare motori, resistenze di riscaldamento e così via. A questo punto della catena entra in gioco un componente simmetrico al convertitore AD: il convertitore DA (Digital to Analog) che ha il compito di convertire le parole binarie del computer in tensioni variabili tra 0 e 5 volt. Di questo argomento comunque ci occuperemo più dettagliatamente in un

Elenco dei componenti

IC1 = ADC0804
IC2 = 74LS32

R1, R2, R3 = 10 KOhm, 1/4 Watt, 1% tolleranza

C1, C2 = 0.1 microFarad
C3 = 150 picoFarad
C4 = 10 microFarad

A titolo indicativo, il costo dell' ADC 0804 si aggira tra le 10000 e le 15000 lire.

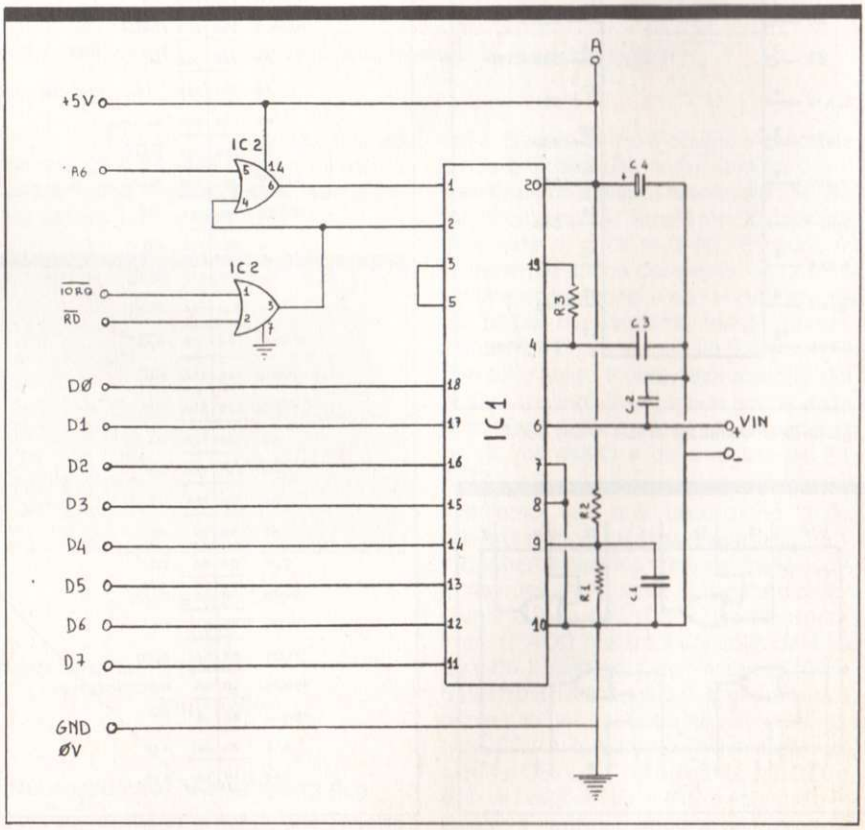


Figura 1: Schema elettrico del convertitore

prossimo numero.

Dicevamo che il convertitore AD misura solo tensioni comprese tra 0 e 5 volt e a questo punto può sorgere spontanea la domanda "ma come faccio io a misurare tensioni molto più piccole, o molto più grandi? e se invece di tensioni devo misurare temerature, luminosita o pressioni?".

La risposta è molto semplice: infatti esistono in commercio innumerevoli tipi di trasduttori per convertire qualsiasi forma di energia fisica in una tensione. Ad esempio con una fotocellula è possibile avere una tensione proporzionale alla luce che la colpisce; certi tipi di conduttori variano la loro resistenza elettrica in funzione della temperatura di lavoro: dalla misura della corrente circolante in essi è possibile risalire alla temperatura.

In questo articolo vi presentiamo un semplice ed economico convertitore AD da collegare allo Spectrum. Si fa uso di un circuito integrato della National Semiconductors, molto sofisticato e complesso.

Tuttavia abbiamo voluto privilegiare la semplicità circuitale, anche se questo significava rinunciare a molte caratteristiche interessanti di questo ADC. In pratica il nostro ADC potrà servire per tutti gli usi descritti finora. Tuttavia per compiti avanzati, come ad esempio il campionamento di segnali musicali fino a 20000 Hertz per farne l'analisi grafica, il nostro progetto mostrerà dei limiti (benchè non abbiamo provato, non escludiamo però che un campionamento possa essere effettuato per frequenze fino a 5000 Hertz).

Il principio di funzionamento del convertitore AD si basa sul concetto di bisezione. Per capire ci aiuteremo con un esempio. Supponiamo che sull'ingresso venga applicata una tensione di 3.54 volts. Internamente all'integrato questa tensione viene confrontata con una tensione di 2.5 volts cioè 5/2. Se il confronto stabilisce che la nostra tensione è maggiore di 5/2, il bit più significativo verrà messo a 1 altrimenti a 0. Nel nostro caso il bit più significativo vale 1. Stabilito questo, ai nostri 3,54 volts ne vengono sottratti 5/2 e il confronto si fa con una tensione di 5/4: cioè 3.54-2.5=1.04 viene comparato con 1.25. In questo caso dato che la nostra tensione è minore, nel secondo bit piu significativo, viene scritto uno 0. Dato che il confronto ha prodotto uno 0, non viene sottratta alcuna tensione

```
OSCILLOSCOPIO

1 BORDER 0: PAPER 0: INK 9
10 CLS
20 FOR F=1 TO 255
30 PLOT F,(IN 191)/2
40 NEXT F
50 GO TO 10

LANCETTA

 1 BORDER 0: PAPER 0: INK 9
10 CLS
20 LET A=IN 191
30 PLOT 128,0
40 DRAW A-128,80
50 PRINT AT 0,0;.1*INT (A/5.1); "    "
60 PLOT 128,0
70 DRAW OVER 1;A-128,80
90 GO TO 20
```

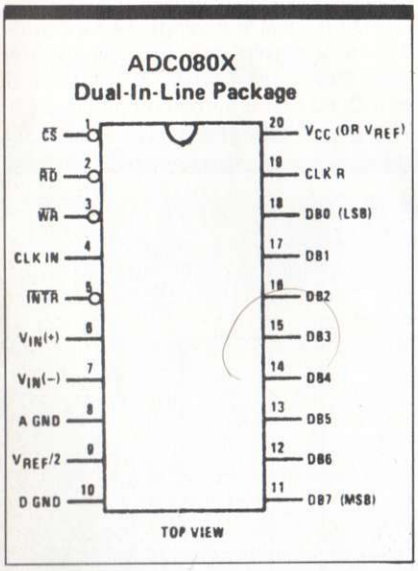


Figura 2a. Piedinatura di IC1 =ADC 0804

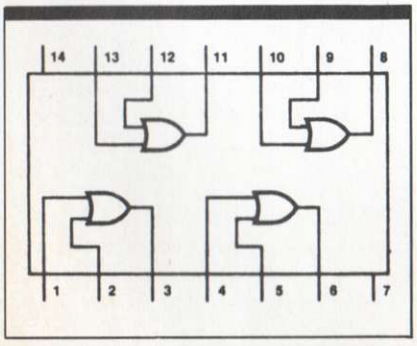


Figura 2b. Piedinatura di IC2 =74LS32

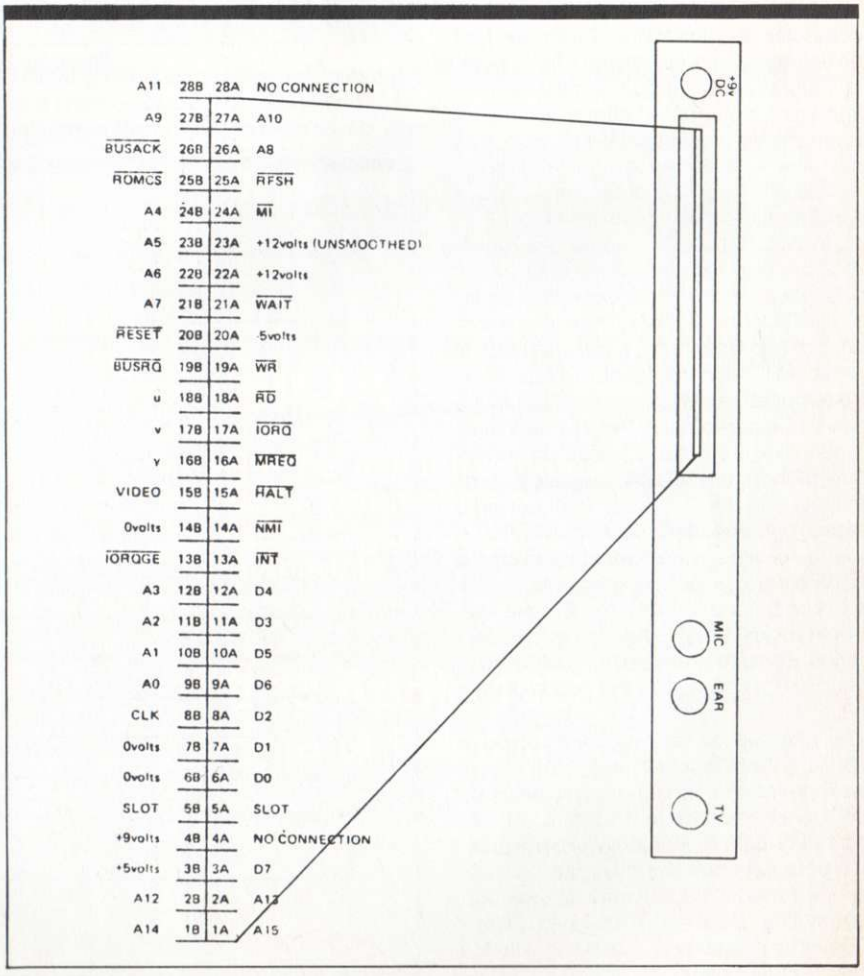


Figura 3. Connettore posteriore dello Spectrum

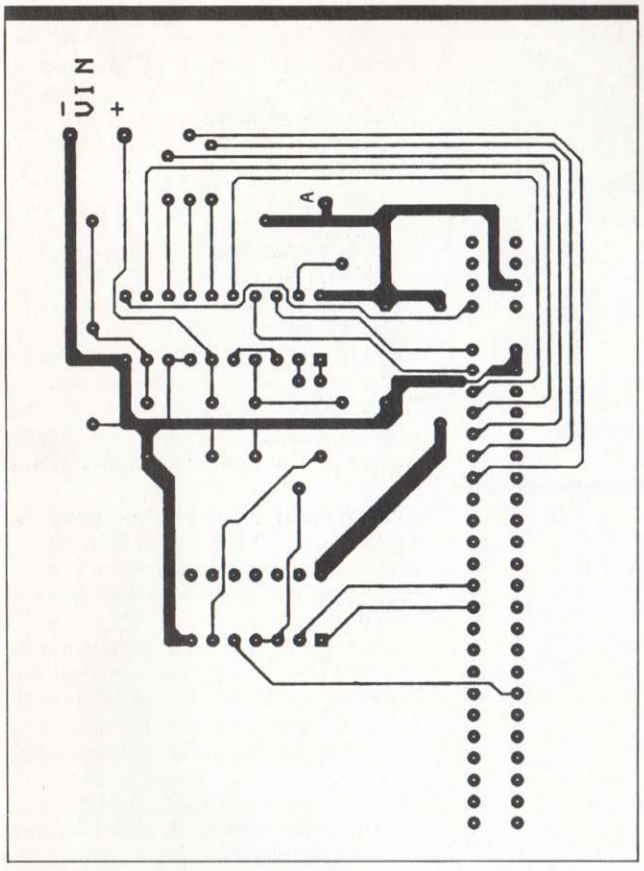


Figura 4a. Circuito stampato dal lato RAME. Le dimensioni reali sono di cm. 9.5x6,7 circa.

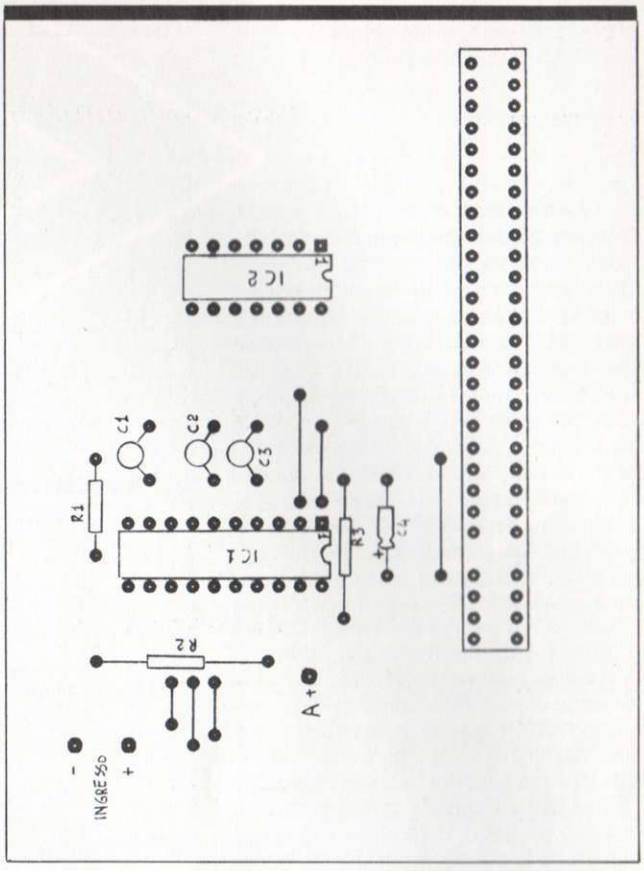


Figura 4b. Circuito stampato dal lato COMPONENTI.

e si esegue un nuovo confronto con 5/8=0.625. In questo caso il risultato del confronto produce un 1 e la nostra tensione viene aggiornata a 1.04-0.625= 0.415. Il processo continua cosi fino ad ottenere gli 8 bit che ci danno il valore della tensione. Nella seguente tabella sono riassunti tutti i passaggi:

V=3.54			
3.54	〈 2.5	? ... BIT 7 = 1 :	V=V-2.5
1.04	〈 1.25	? ... BIT 6 = 0	
1.04	〈 0.625	? ... BIT 5 = 1 :	V=V-0.625
0.415	〈 0.3125	? ... BIT 4 = 1 :	V=V-0.3125
0.1025	〈 0.15625	? ... BIT 3 = 0	
0.1025	〈 0.078125	? ... BIT 2 = 1 :	V=V-0.078125
0.02437496	〈 0.0390625	? ... BIT 1 = 0	
0.02437496	〈 0.01953125	? ... BIT 0 = 1	

Questi 8 passaggi forniscono il valore BIN 10110101 cioè 181 decimale. Dato che il massimo valore che possiamo leggere in uscita dal convertitore AD è 11111111 cioè 255 decimale che indica 5 volt, per sapere a quanti volt corrisponde il valore 181 dobbiamo fare la seguente proporzione:

181:255=x:5

Il valore della tensione è quindi dato dalla formula

$$\frac{\text{valore letto} * 5}{255}$$

o equivalentemente

$$\frac{\text{valore letto}}{51}$$

Nel nostro caso 181*5/255=3.549

I più accorti avranno notato che il nostro misuratore dà un errore di quasi un centesimo di volt rispetto alla misura reale di 3.54. Questo errore è dovuto al fatto che per esprimere tutte le tensioni tra 0 e 5 volt abbiamo a disposizione solo 256 valori (8 bit) per cui l' errore medio delle misure sarà di circa lo 0.4%. Esistono in commercio anche convertitori a 12 o 16 bit che consentono errori enormemente più piccoli ma riteniamo che per un uso hobbistico uno scarto dello 0.4% sia più che tollerabile. Inoltre la precisione del nostro convertitore dipende anche dalla precisione della tensione di alimentazione (5 Volt esatti) e delle resistenze R1 e R2.

Veniamo ora alla descrizione dello schema elettrico. Se osservate la figura 1 e lo schema elettrico noterete che le uscite dati dell' ADC sono collegate direttamente al bus dati (D0-D7) dello Spectrum. L' ADC prende il controllo del bus quando il piedino 1 dell'integrato (Chip Select) è a livello logico 0. Il tipo di logica contenuta nell circuito integrato denominato IC2 (un quadruplo OR) fa si che sul piedino CS di IC1 sia presente uno 0 se e solo se i segnali RD, IORQ e A6 sono a 0. Questo si verifica durante le operazioni di lettura (RD=read=0) da una porta di IO (IORQ=Input Output ReQuest=0) che

abbia il sesto bit di indirizzo a 0 (A6=0). In pratica questo avviene durante le istruzioni BASIC contenenti un

IN (BIN 10111111)

o equivalentemente

IN 191

Il resto della circuiteria (resistenze e condensatori) serve di supporto per IC1. Una piccola nota va dedicata a C1 e C2. Questi due condensatori da 0.1 microfarad sono stati inseriti per proteggere il circuito da segnali spuri causati da una cattiva realizzazione o da cavi di input troppo lunghi eccetera, cioè, per utilizzare una terminologia precisa, per filtrare il rumore della misurazione. Tuttavia questi due componenti causano un inconveniente: rendono cioè "lento" il sistema rispetto alle variazioni dell'input. Questo significa che se l'input passa bruscamente da 4 a 1 volt, l'ADC ci segnalerà questo passaggio qualche millesimo di secondo dopo che è avvenuto.

Sempre sull'input vale una raccomandazione molto importante: per ridurre all'osso i componenti utilizzati, la nostra tensione entra direttamente nel IC1: non esiste nessun tipo di protezione, quindi state attenti a non immettere sull'ingresso VIN tensioni superiori a 5 volt o negative.

Se utilizzerete questo convertitore in unione a programmi scritti in linguaggio macchina potrebbe verificarsi un inconveniente dovuto alla frequenza con la quale fate le letture sull'ADC. Infatti, sempre per ridurrre il progetto all'osso, non esiste un segnale in uscita dal convertitore che ci segnali quando questo ha letto il valore in input, , effettuato le comparazioni necessarie ad elaborare il valore e presentato correttamente i dati sulla porta. Data la grande velocità del linguaggio macchina potrà capitare di fare una richiesta di lettura, prima che si sia conclusa l'operazione riguardante una richiesta di lettura precedente. In questo caso i valori letti non saranno reali e sarà quindi necessario inserire dei loop di ritardo tra una lettura e l'altra.

La realizzazione pratica del convertitore non dovrebbe presentare grosse difficoltà: dato l'esiguo numero di componenti si può anche fare uso di una basetta preforata che consente di risparmiare il tempo necessario alla realizzazione di un circuito stampato. Tuttavia per coloro che preferissero questa seconda soluzione vi presentiamo anche il disegno di un circuito stampato sul quale realizzare il nostro progetto. Se proprio di elettronica non ne capite niente e non avete mai preso in mano il saldatore potete sempre farvi aiutare da un amico più esperto al quale in cambio potrete presentare la ragazza più carina che conoscete (le ragazze

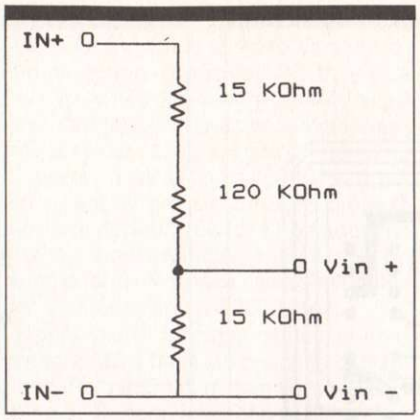


Figura 5. Collegamenti del potenziometro da 10 kOhm per il collaudo del convertitore

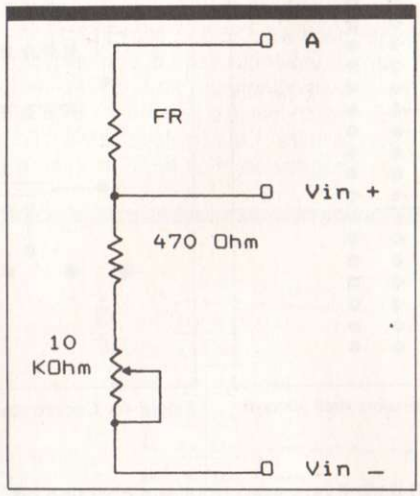


Figura 6. Esempio di applicazione per la misura delle intensità luminose: FR è una fotoresistenza; il potenziometro da 10 kOhm serve per stabilire la sensibilità. Data l'estrema semplicità di quest'applicazione, la risposta alla luce non sarà lineare. Per realizzare un misuratore di temperature, è sufficiente costituite la fotoresistenza con una resistenza NTC

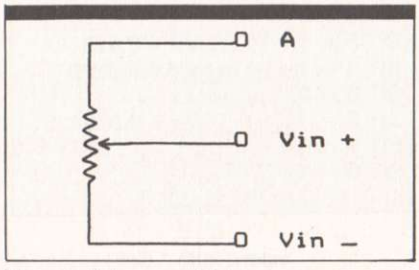


Figura 7. Divisore per 10 della tensione applicata agli ingressi IN+ e IN- (su IN+ applicare il positivo e su IN- il negativo della tensione da misurare

desiderose di esplorare il fantastico mondo della conversione AD possono invece rivolgersi direttamente qui in redazione...).

Per tutti, raccomandiamo caldamente l'uso di zoccolini su cui montare i due circuiti integrati, questo per evitare di distruggerli indugiando troppo con un saldatore troppo potente sui piedini degli IC che, benchè sufficientemente robusti sono abbastanza sensibili a eccessi di calore. Inoltre, in caso di guasto, sostituire un IC sarà semplificato.

Se realizzate le versione su circuito stampato, non dimenticatevi di montare i 6 ponticelli realizzati con degli spezzoni di filo.

Per la connessione allo Spectrum esistono in commercio appositi connettori ai quali, oltre agli 8 bit di dati vanno collegati i segnali di IORQ, A6, RD (oltre naturalmente alla massa e all'alimentazione). Le connessioni di questi segnali sul connettore dello Spectrum sono riportate nella figura 3.

Il collaudo del nostro convertitore è abbastanza semplice. Collegate ai pin di ingresso un potenziometro da 4.7 o 10 KOhm secondo lo schema mostrato in figura 5.

Dopo aver verificato con la massima attenzione che tutti i componenti del converter sono stati montati correttamente, CON LO SpectRUM SPENTO si collega il nostro convertitore al connettore posteriore. Quindi provate a dare tensione. Se non succede niente di strano e il computer si accende come di consueto, va tutto bene. Se invece, la procedura di inizializzazione dello Spectrum non ha luogo correttamente, spegnete subito il computer e ricontrollate il montaggio perchè c'è qualche errore. Supponendo che tutto proceda bene, inserite il seguente programmino di prova

```
10 PRINT AT 0,0; IN 191;" "
20 GO TO 10
```

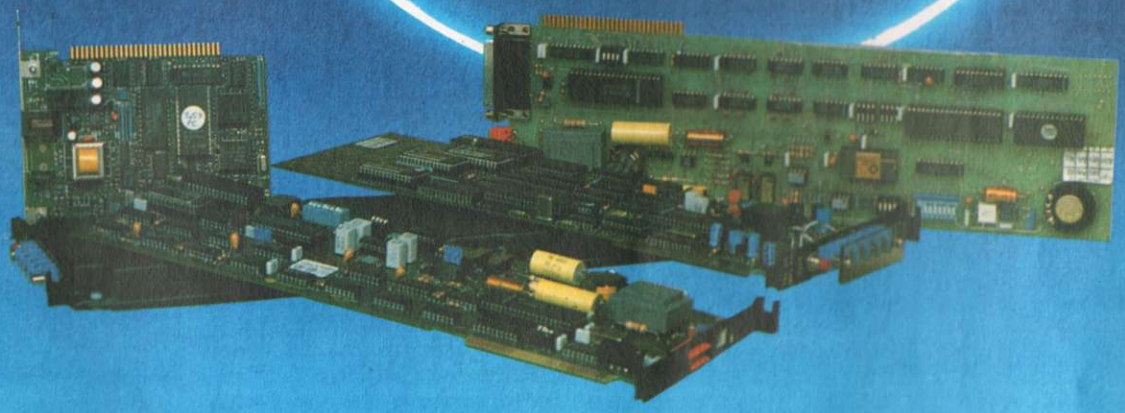
e date RUN.

girando il potenziometro da una parte all'altra, dovrete leggere tutti i valori tra 0 e 255; un eventuale oscillazione di più o meno una unità è normale.

A questo punto il collaudo è finito e il vostro convertitore è pronto per funzionare, potete scollegare il potenziometro e inviare ai piedini di ingresso la tensione che vorrete misurare. A titolo indicativo vi forniamo alcuni semplicissimi schemi. Uno indica come realizzare un divisore per 10 per poter leggere tensioni tra 0 e 50 volt e l'altro propone una soluzione semplicissima per collegare una cellula fotoelettrica o un NTC (resistenza variabile in funzione della temperatura) al convertitore per misurare luci e temperature.

Come esempio forniamo anche due programmini per avere due diversi tipi di visualizzazione della misura letta: uno simula un classico strumento a lancetta e l'altro imita il funzionamento di un oscilloscopio.

nel
segno
digicom



Tanti modem a scheda...

M3-12PC per trasmissione di dati a 300 bps full-duplex, 1200 bps half-duplex e 1200/75 bps full-duplex (Videotel) con adattamento automatico al tipo di modulazione del modem remoto e chiamata telefonica automatica tipo Hayes.

SNM-12PC per trasmissione di dati a 300 bps full-duplex e 1200 bps full-duplex con adattamento automatico al tipo di modulazione del modem remoto e chiamata telefonica automatica tipo Hayes.

M24-24PC il modem professionale polifunzionale per trasmissione di dati in full-duplex a 300, 1200, 1200/75 e 2400 bps ed in half-duplex a 1200 bps, con adattamento automatico a standard di modulazione e baud-rate del modem remoto. Correzione degli errori di linea secondo CCITT V41 o MNP liv. 3 e chiamata telefonica automatica secondo CCITT V25Bis e Hayes.

...per comunicare con il personal computer

Studio Office Automation

digicom s.p.a.

21013 Gallarate (Varese) - Via Curioni, 14 - Tel. (0331) 78.34.09 - Tlx 333679 digico I
Uffici regionali: 00145 ROMA - Via C. Colombo, 436 - Tel. (06) 51.34.628 - Tlx 625481 digiro I
40122 BOLOGNA - Via Lane, 116 - Telef. (051) 55.87.09

JL ATARI

IL MENU' DELL'ST

di Valerio Ferri

Un chiaro esempio su come funziona la gestione dei menù in Gfa-Basic

Tutti i programmi, generalmente vengono guidati nella loro esecuzione tramite classici "input". Nell'Atari St vi è un'alternativa che dà la possibilità di sostituire il preistorico metodo con finestre a discesa. Oltre a rendere più naturale la comunicazione tra uomo e computer, icone e finestre riescono a donare un certo stile di professionalità al programma stesso. Queste, inoltre, sono preferite grazie alla particolare facilità con cui possono essere gestite.

Il listato, semplice e conciso, chiarirà sicuramente tutti i dubbi a chi nutriva particolare interesse verso questo tipo di applicazione del Gfa-Basic.

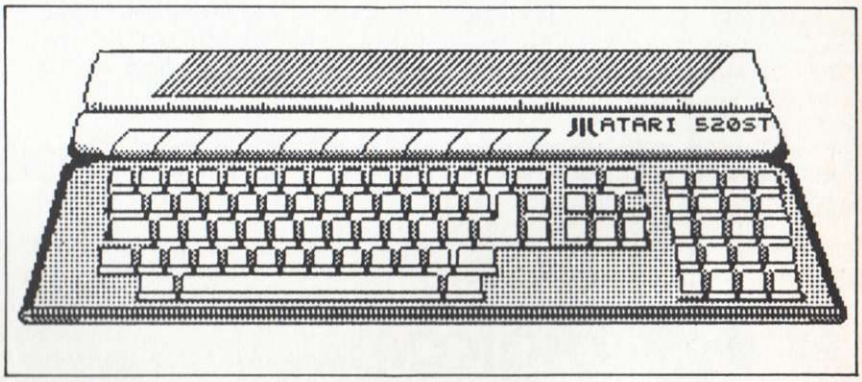
Le procedure sono principalmente due, una denominata "Init", la quale provvede a cancellare lo schermo, visualizzare il menù (menu scelta$()) e definire il tipo di mouse (nel nostro caso la "manina"). L'altra, denominata "Main", è quella a cui il programma dovrà fare riferimento durante il funzionamento. Quest'ultima procedura, intercetta le chiamate e richiama a sua volta le routines necessarie.

Le variabili di maggior importanza sono: "vie" e "scelta". La prima, numerica, viene usata per conoscere a priori il numero di comandi possibili, la seconda, ("scelta") alfanumerica, funge da vettore. In essa sono memorizzate le singole stringhe da visualizzare nelle finestre.

Una volta in esecuzione il programma, con il mouse si seleziona una funzione, premendo come abitudine il tasto sinistro a scritta in "reverse". La procedura "main" riceve attraverso "menu(0)" un numero. Questo corrisponde alla posizione dell'opzione desiderata all'interno del vettore "scelta". In questo modo si conosce quale funzione è stata selezionata.

Molti di voi si saranno chiesti per quale motivo sia stato affrontato un argomento così specifico. Ebbene, tempo fa un lettore di Napoli, recentemente un abbonato di Vercelli, ci scrissero chiedendo chiarimenti sulla gestione dei menù nell'Atari St. Inutile dire che ci siamo sentiti in dovere di pubblicare un semplice esempio di questa non poi difficile applicazione.

(List a pag. XXIV)



STARGLIDER

di Roberto Zubani

Un simulatore di volo a bordo di un caccia stellare

Benvenuti a bordo di un "AGAV", ovvero l'Airborne Ground Attack Vehicle, il più rivoluzionario dei nuovi caccia biposto disegnato e costruito dalle Draziw Industries, leader delle fabbriche di armamenti di Novenia da più di ottocento anni.

Una volta entrati nella cabina, controllate tutti i comandi presenti, chiudete lo sportello e preparatevi al decollo.

Al centro della consolle di controllo c'è lo "scanner" a lungo raggio, e poco sotto appaiono le coordinate del pianeta che state sorvolando.

Oltre ai più moderni sistemi di trasmissioni radio, l'Agav dispone di un sofisticato computer di bordo che vi dà, in tempo reale, tutte le informazioni necessarie al volo ed al combattimento.

Anche il controllo degli stabilizzatori ai neutroni, e il compito di assicurare un ottimo funzionamento dei motori al plasma è affidato al calcolatore.

Sulla sinistra sono collocati gli indicatori sullo stato dello scudo molecolare neutralizzante e delle cellule laser, che possono essere ricaricate tramite i Silos equipaggiati con rigeneratori.

Nella parte inferiore, i tecnici hanno sistemato l'altimetro digitale e l'indicatore dei missili a disposizione.

Sulla destra è allocato il contatore dell'energia globale che serve ad alimentare i propulsori. La velocità massima raggiungibile, visualizzata tramite l'indicatore laterale, è (fate voi i calcoli..) di 2550 Urads (1 Urad = 2,37218 Km/h).

Inoltre, su questo modello sono istallati due indicatori gemelli per le virate e per lo stato energetico dei propulsori, di cui però il pilota non deve preoccuparsi, dato che ad essi pensa il computer di bordo.

Infine, sopra il "display" principale è indicata la rotta in gradi, in relazione alla posizione della stella Irralya.

Dalla teoria alla pratica

Volare con l'Agav è molto semplice, sia che si voglia sorvolare il territorio raso terra o raggiungere le quote più alte. E' facile anche atterrare, riducendo la velocità a zero.

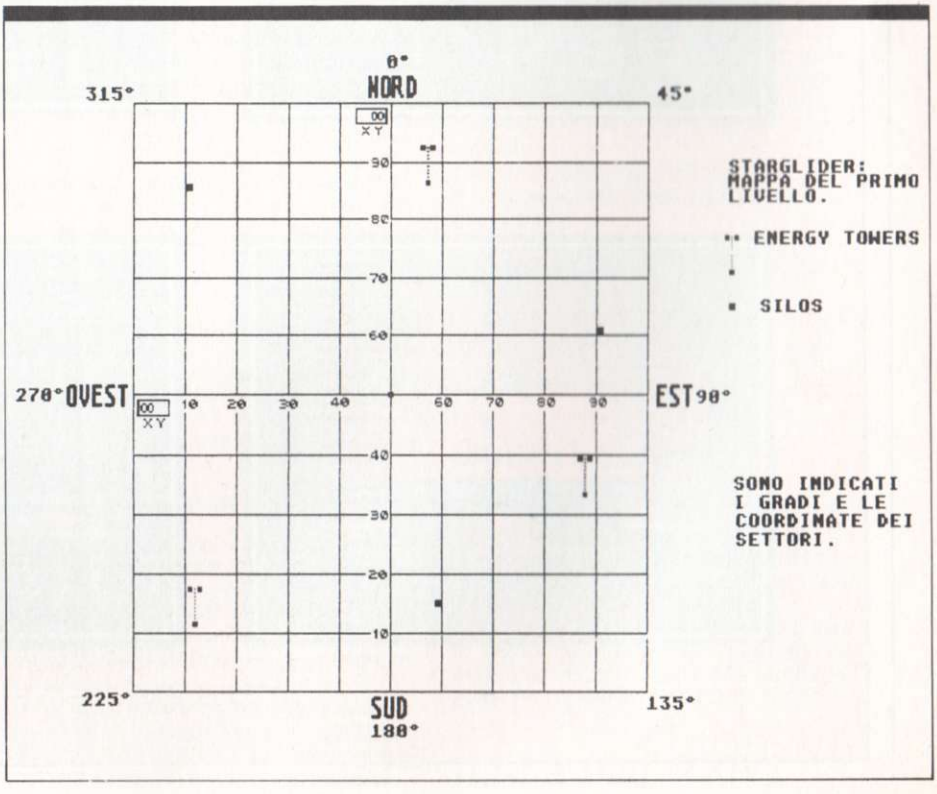
Grazie ai potenti motori che il velivolo dispone, esso è in grado di superare tutti i veicoli alleati che viaggiano a velocità sub-luce, nonchè i mezzi Egron e Aruloid attualmente in servizio.

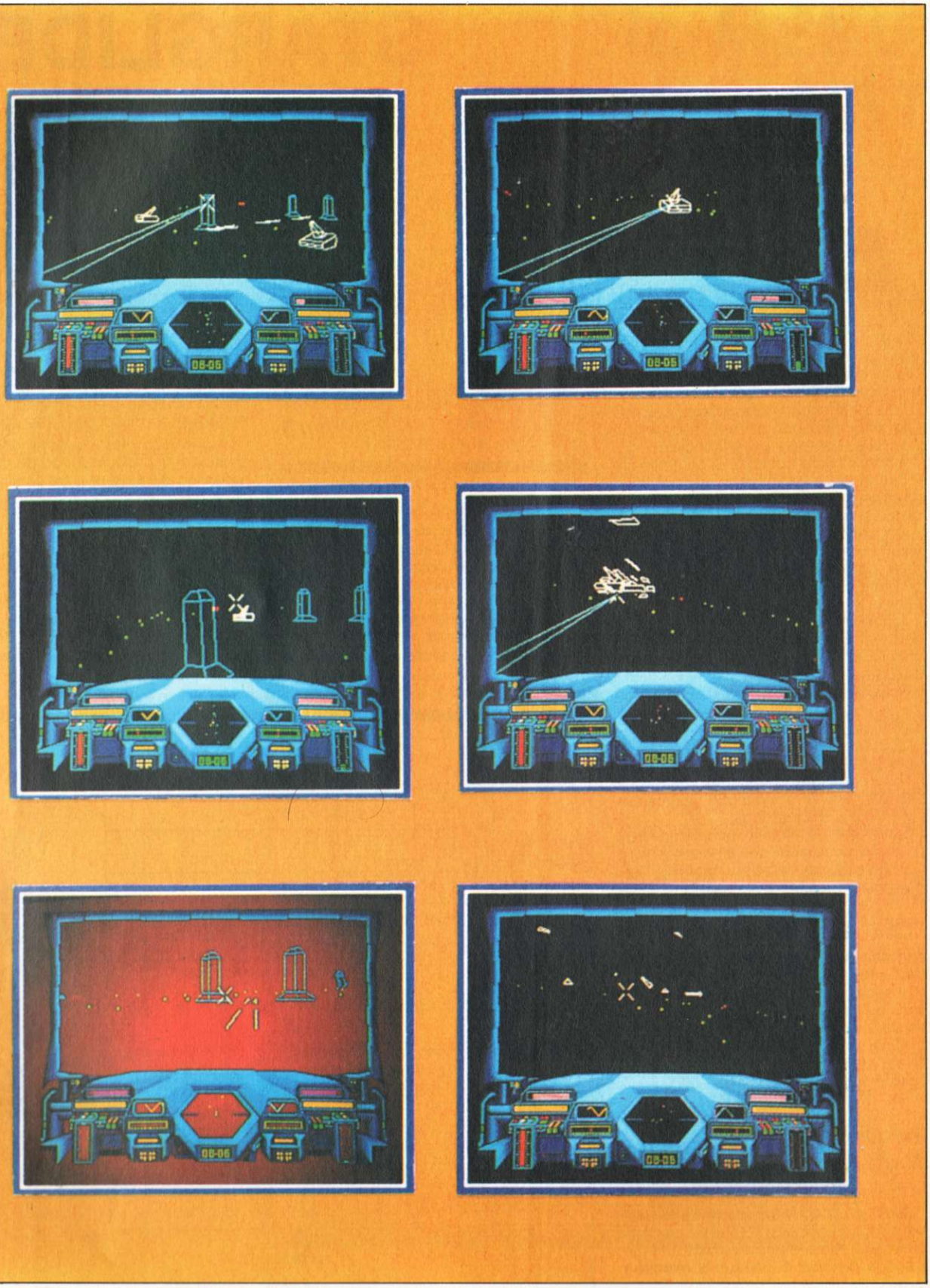
La massima virata è di circa 45 gradi, ed anche alla massima velocità ci vuole pochissimo tempo per invertire la rotta.

Questo favoloso aereo, dispone di carrelli che consentono l'atterraggio sui territori più ostili, inclusa la lava fusa, le rocce acide e le distese di funghi di Erialc. Tuttavia, la velocità del veicolo deve essere zero, se non si vuole correre il rischio di rovinarne la parte inferiore, perdendo oltretutto energia quando gli scudi cercheranno di opporsi all'attrito.

E' possibile rifornirsi posizionando la punta del velivolo in modo da tagliare il raggio ad altissima energia che collega le "Energy-Towers". La tabella riportata, indica la posizione delle torri al primo livello.

Come accennato in precedenza l'Agav monta un sistema laser Sapphire II, a cui sono dedicate speciali cellule energetiche disposte nella punta del veicolo. I laser sono posti sotto ciascuna ala e sparano simultaneamente due raggi. Il sistema di mira computerizzato può essere reso





indipendente dal volo, consentendo una maggior velocità e accuratezza nel puntamento. La potenza dei lasers è tale che sono sufficienti dai 4 ai 6 colpi per distruggere ogni tipo di veicolo nemico.

Gli Agav, che fanno parte delle flotte stellari da sbarco, sono equipaggiati con missili ai protoni a corto raggio MK 14. Questi missili dispongono del sofisticato sistema di autoguida Vidimon. Esso permette al pilota di quidare direttamente il missile sul bersaglio tramite una telecamera montata sulla punta di quest'ultimo.

I comandi disponibili:

•ESC: QUIT
•BACKSPACE: PAUSA
•S/Q: SUONO ON/OFF
•F: MIRINO MOBILE O FISSO
•I: INFORMAZIONI DAL COMPUTER DEL SILO
•K: DEFINIRE I TASTI DA USARE
•L: LANCIO MISSILE
•SPACE: LANCIO DAI SILOS
•FRECCE: MOVIMENTO E VARIAZIONI DI QUOTA
•X/C: VIRATA
•;/⟩: VARIAZIONI DI QUOTA
•SPACE/?: ACCELERAZIONE E DECELERAZIONE
•RETURN: LASER

Si può variare la velocità muovendo il mouse avanti e indietro con il bottone destro premuto. Il bottone sinistro controlla invece i lasers. Ogni 10000 punti si passa al livello successivo, in cui la posizione dei silos e delle torri di energia cambia, e aumenta il numero dei nemici.

Per avere una descrizione di quest'ultimi e sufficiente interrogare il computer situato nei silos di rifornimento.

Conclusioni

Con Starglider, potrete provare l'emozione di volare sul veicolo mentre sotto di voi scorrono velocissime immagini del territorio che state sorvolando. Vi offre, infatti, la possibilità di sedervi di fronte a quello che in pratica è un simulatore di volo a bordo di un caccia stellare.

I tipi di bersaglio che potrete incontrare non si contano, dalle velocissime navette da battaglia ai lenti e impenetrabili bipedi d'assalto, incredibili robots che ricordano i quadropodi imperiali del film "L'impero colpisce ancora.".

Sorprendente è la presentazione iniziale, costituita da una "super" schermata grafica, accompagnata da un ottimo brano digitalizzato. Voci campionate vi avvertono quando l'energia degli scudi o globale è bassa e voi rischiate di venir distrutti nel giro di pochi attimi.

Non ci resta che augurare a tutti un buon divertimento!

ZETAGRAF

di Lorenzo e Roberto Zubani

Nuove procedure per ottimizzare il vostro C.A.D. personalizzato

Come promesso, ecco alcune procedure da aggiungere al programma "Zetagraf" precedentemente pubblicato. Prima di tutto un "riassunto" per tutti coloro che hanno mancato il numero contenente questo programma grafico per Atari St, adatto a funzionare in ambiente monocromatico e scritto in GfA basic.

Lo scopo è quello di disporre di un programma modulare, al quale aggiungere continuamente nuovi algoritmi sottoforma di procedure, non solo da noi pubblicate, ma anche da voi create per particolari necessità.

Le procedure di questo mese sono parecchie, e prima di descriverle una per una ricordo in che modo vanno aggiunte al programma principale.

Osservando il listato noterete che l'ultima parte è stata qui ripetuta, questo perchè contiene alcune modifiche che dovrete apportare al vostro programma confrontando i due gruppi di linee.

Fatto ciò sarà sufficiente aggiungere in coda al programma tutte le procedure una dopo l'altra e per utilizzarle basterà togliere i segni "-" davanti ai nomi relativi nei Data dei menù.

Tra i vari sottoprogrammi ve ne sono due chiamati Standard e Utente. Il primo serve a riportare allo stato di default i parametri grafici (es. stile linee, spessore, modo grafico, colore, ecc..) ed il secondo ad istallare quelli da voi scelti. Il vantaggio è quello di evitare che le modifiche fatte, ad esempio allo spessore delle linee, rovinino il funzionamento delle altre procedure.

Quindi si tratta di usare la chiamata ⟨Gosub Utente⟩ quando inizia per esempio la procedura per tracciare linee, e ⟨Gosub Standard⟩ quando essa finisce.

Esaminando le procedure

• ⟨NOTE⟩: quante volte lavorando vi sarà capitato di voler annotare dei dati senza avere a disposizione un foglio su cui scriverli? ebbene, con questa procedura disporrete di un "block notes" visualizzato su schermo che vi consentirà di risolvere questo problema.

• ⟨SPEC⟩: si tratta semplicemente della possibilità di ottenere immagini speculari secondo varie direzioni, con essa potrete costruire simpatici effetti caleidoscopici.

• ⟨ARC⟩: talvolta può servire il tracciamento di archi di predefinite dimensioni, ecco allora che abbiamo pensato a un algoritmo che, ricevendo i gradi iniziali e finali esegua tale funzione.

• ⟨POL⟩: genera poligoni regolari di dimensioni variabili, per il momento solo pentagoni, esagoni e ottagoni.

• ⟨ESTR⟩: consente la scelta degli estremi delle linee in uso, a freccia, arrotondati ecc..

• ⟨DEFL⟩: genera un tipo di linea definito dall'utente mediante l'immissione di una sequenza di "1" e di "0".

• ⟨DEFP⟩: crea una pattern posizionando un quadratino mobile su una zona del video in cui essa sia stata prima disegnata. Si consiglia di usare questa funzione insieme a ⟨LENT⟩ per ottenere risultati migliori.

• ⟨MIS⟩: stampa sul video in tempo reale le coordinate in pixel di un cursore mobile a croce.

• ⟨ASCII⟩: visualizza in un rettangolo i codici Ascii dei caratteri previsti nel set disponibile.

• ⟨LENT⟩: gestisce una semplice e pratica lente di ingrandimento.

• ⟨RIGH⟩: questa opzione consente l'uso di righelli in scala centimetrica.

Ulteriori informazioni

Le successive procedure, presenti nell'ultima parte del listato, sono di supporto ed essenziali al funzionamento del programma, si raccomanda quindi la massima cura nel digitarle.

Per qualsiasi problema inerente al listato, telefonate o scrivete in redazione oppure a: Lorenzo Zubani - Via Monte Popera 16/12 Milano (tel. 02/513098).

(List a pag. XXIV)

TANTI BUONI MOTIVI PER ABBONARSI A

VR VIDEOREGISTRARE

12 NUMERI AL PREZZO DI 10 solo 45.000 lire invece di 54.000 lire

•

PREZZO BLOCCATO per tutta la durata dell'abbonamento

•

SICUREZZA di non perdere neanche un numero

•

COMODITÀ di ricevere la propria rivista preferita a casa

•

VR VIDEOREGISTRARE

•

COSA STATE ASPETTANDO?

VIDEO A 160 COLONNE

di Valerio Ferri

Un semplice algoritmo che permette di scrivere e visualizzare testi in pagina grafica

Chiunque abbia avuto la necessità di visualizzare una scritta in pagina grafica, avrà senza dubbio incontrato parecchie difficoltà nella stesura di un'algoritmo semplice e nello stesso tempo veloce, che risolva tale problema.

Una simile procedura, ricevendo in ingresso una stringa, dovrebbe disegnarla in pagina grafica carattere per carattere. Ma non è così sempice, ogni carattere infatti, deve essere prima convertito in "shape" (matrice di pixel) e successivamente indirizzato sul video punto per punto.

Esistono comandi che a loro modo risolvono questo problema, ma purtroppo sono limitati dalla dimensione del carattere e dallo stile dello stesso.

Ciò che vi proponiamo non è sicuramente il massimo che si possa ottenere in fatto di velocità, ma quello che principalmente conta è il risultato, e questo è garantito.

Il linguaggio usato per lo sviluppo del programma è il diffusissimo Gfa-Basic. Più idoneo sarebbe stato il linguaggio "C" o addirittura il codice macchina ma, data la complessità di tali linguaggi si sarebbe ottenuto un'algoritmo poco chiaro.

Il compito del programma è di visualizzare un file testo in pagina grafica con caratteri definiti dall'utente.

L'utilizzo di questo programma è molto semplice, infatti basta rispondere alla domanda "Nome del file da leggere: " proposta dal computer, con il nome del file da esaminare. L'unica condizione da rispettare per il corretto funzionamento è che il file in "input" sia in formato Ascii (per intenderci simile al file "DESKTOP.INF").

Una volta in esecuzione, il programma inizia a scrivere sul video, carattere per caratere, il contenuto del file con una risoluzione di 160x66 caratteri. Usando caratteri ridefiniti con matrice di 4x6 pixel si ottiene infatti una risoluzione di 160 colonne per 66 righe dato che l'Atari St dispone di una grafica di 640x400 punti.

Con alcune modifiche i più esperti potranno modificare a piacimento sia la grafica dei caratteri, sia la dimensione degli stessi. Con queste modifiche e un poco di fantasia sarà poi un gioco da ragazzi sviluppare un mini "desktop publishing".

Il programma procedura per procedura

Come si può notare dal listato, il programma è molto semplice. Ma una spiegazione alle varie procedure sarà sicuramente utile.

- Getkey: attende che venga premuto un tasto.
- Hprint: converte e disegna, carattere per carattere la stringa "X$" alle coordinate X, Y. L'esempio che segue il listato dove viene scritta la stringa "premi un tasto" alle coordinate X=140, Y=65 ne chiarirà il concetto.
- Init: inizializza le variabili e seleziona il file da esaminare. Prepara la matrice contenente i dati relativi ai caratteri da "plottare".
- Readchar: preleva un dato dal file di input.
- Ascii: converte il dato in ingresso "A$" in codice relativo alla matrice in precedenza definita.
- Contret: controlla se il codice in ingresso è un "ritorno carrello".
- Plotchar: è la procedura principale, disegna il carattere "CH" in pagina grafica alla posizione X, Y stando ai valori contenuti nella matrice. E' possibile cambiare a piacimento lo stile dei caratteri che vengono visualizzati. Infatti, modificando le linee che contengono i "data", invertendo a scelta gli zeri in uno e viceversa si varia la forma del carattere sul video.

Due esempi di "output" ottenuti con il programma "160x66" (uno dei quali è stato ottenuto con il conosciutissimo "Desktop.inf"), sono riportati nelle figure a lato.

(List a pag. XXIX)

L'ASSEMBLER

a cura di Marco Silvestri

Stack, I/O, Interrupt

Lo Stack

Programmando in Assembler ci si rende presto conto del fatto che i registri sono un mezzo molto limitato per contenere i dati sui quali si lavora, perchè oltre ad essere pochi sono spesso modificati dalle routines del BIOS o devono essere utilizzati nei programmi per compiti specifici, come A per le operazioni logiche o BC ed HL per una CPIR.

Inoltre spesso succede di dover scambiare i dati fra due diversi registri senza avere a disposizione l'istruzione necessaria: è il caso, per esempio, dell'inesistente *LD HL,AF*.

Per offrire una soluzione a tutti questi problemi, che sono frequentissimi e decisivi nella programmazione in linguaggio macchina, il microprocessore mette a disposizione un particolare strumento, lo **Stack**, tipica struttura LIFO (Last In, First Out; l'ultimo che entra è il primo che esce); come per una pila di piatti, l'ultimo piatto posto sulla pila sarà anche il primo ad essere tolto a meno di non voler rovesciare il tutto!.

Per comprendere bene il funzionamento dello stack è necessaria un po' di immaginazione, poichè si tratta di qualcosa di inconsueto dal punto di vista concettuale per chi è abituato a programmare in Basic.

Dopo aver compreso il concetto che sta alla base di questo strumento, passiamo ad esaminare con più cura quella che è la sua effettiva organizzazione: intanto occorre precisare che lo Stack si estende all'indietro nella memoria ovvero in direzione della locazione 0, come dire che la nostra pila ha il primo piatto alla locazione nn, il secondo a nn-2, il terzo a nn-4, e così via.

Ok, ma chi tienè il conto? C'è un registro apposito a 16 bit di nome **SP** (Stack Pointer) che contiene l'indirizzo della cima della pila e che viene decrementato o incrementato di due ad ogni operazione che rispettivamente aggiunga o tolga dati dallo Stack.

Le istruzioni che tipicamente aggiungono o tolgono "piatti" allo Stack sono **PUSH rp** e **POP rp**.

Esemplifichiamo una **PUSH HL**:
SP punta all'indirizzo FFFFh (65535 decimale): dopo PUSH HL, il contenuto di H sarà scritto in FFFEh (65534), il contenuto di L in FFFDh (65533): SP verrà decrementato di due e varrà quindi FFFDh.

La **POP HL** (o la POP qualsiasi altra cosa) effettuerà il percorso inverso della PUSH HL: (FFFDh) in L, (FFFEh) in H, SP=SP+2= FFFFh.

E' anche possibile agire direttamente sul registro SP con le solite LD, INC e DEC, che ci permettono di spostarci abbastanza liberamente nell'area dello Stack. Ma riservando la dovuta attenzione.

Lo Stack Pointer viene utilizzato anche in un altro caso:
quando si esegue una CALL o una RST, l'indirizzo che segue detta istruzione (in effetti l'indirizzo della prossima istruzione da eseguire al ritorno dalla Subroutine) viene anch'esso salvato nello Stack e quindi SP segue le regole già enunciate.

Quando si incontra la RET della subroutine, l'indirizzo prima salvato viene ripescato dallo Stack e riscritto nel registro PC (Programm Counter) e quindi, come per magia, il programma continua dall'istruzione successiva la CALL (o RST).

Ora, vi immaginate cosa può succedere se nella subroutine c'era una PUSH senza la relativa POP? Succede che in PC viene ricaricato non l'indirizzo di ritorno, bensì quello che abbiamo infilato con la PUSH! Avete comunque una probabilità su 65536 di ritornare al punto giusto! Buona fortuna!

E' quindi buona regola che il numero di PUSH eseguiti sia uguale a quello dei POP, così come CALL e RST abbiano i relativi RET. A meno che non sappiate cosa state facendo.

Lo stesso discorso si applica anche alla chiamata da Basic; in questo caso, più o meno, l'interprete salva sullo Stack l'indirizzo dell'istruzione Basic che sta eseguendo prima di saltare alla subroutine e dopo la RET si aspetta di ritrovarvi l'indirizzo giusto: se questo non accade potete facilmente immaginare i problemi che nel 99 % dei casi vengono provocati: il Program Counter salta chissà dove e si ha il solito Crash.

Le Istruzioni di Input/Output

Lo Z80 non ama isolarsi e per questo motivo i progettisti hanno predisposto un modo con cui il microprocessore possa interfacciarsi con il mondo esterno: le porte di Input e Output.

Una radio da ascoltare?

NOI DICIAMO JAY,

Massimo Capozzi & Pick Up - Foto Ghidini

DEEJAY®

anche tu!

Eccoci qui, siamo le voci di Radio Deejay! Da sinistra verso destra puoi vedere Maurizio Desinan, Ronny Hanson, Claudio Cecchetto alla guida, Albertino in alto, Gerry Scotti e Marco Biondi al centro, Linus e Tony Severo in basso, David Lee Stone e Amadeus. Radio Deejay suona anche nella tua città: cercala sulla banda FM.

RADIO DEEJAY NETWORK

Concessionaria per la pubblicità nazionale
Telefono 02/4981841

SPER

Via Giovanni De Alessandri, 11 - 20144 Milano
Telefax 4390724

Routines BIOS per la gestione del PSG

Le routines BIOS del sistema MSX qui descritte controllano il Generatore Sonoro Programmabile (PSG), circuito ampiamente descritto in una precedente serie di articoli di **Personal Computer**.

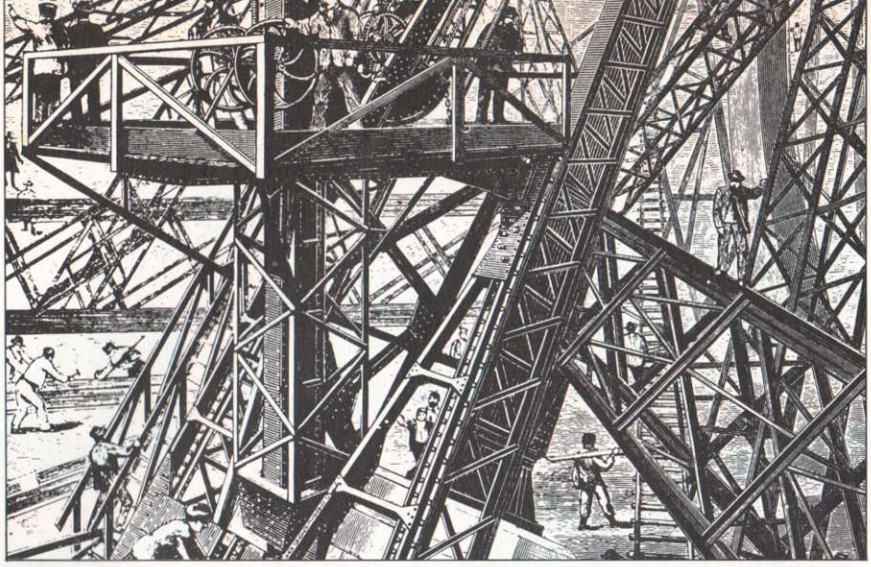
Nome:	GICINI
Indirizzo:	&H0090 - 00144
Funzione:	Inizializza il PSG, annullando eventuali suoni in esecuzione e sopprimendo le code sonore in memoria.
Input:	E' necessario disabilitare le interruzioni prima della chiamata.
Output:	Non viene restituito alcun valore significativo.
Alterazioni:	I registri Z80 non vengono modificati, ma vengono azzerate le variabili di sistema: MUSICF (&HFB3F), PLYCNT (&HFB40), VCBA/VCBA+4 (&HFB41/FB44), VCBB/VCBB+4 (&HFB66/&HFB6A), VCBC/VCBC+4 (&HFB8B/&HFB8F).

Nome:	WRTPSG
Indirizzo:	&H0093 - 00147
Funzione:	Scrive un valore in uno dei registri del PSG.
Input:	Il numero del registro in cui scrivere, compreso tra 0 e 13 quindi, deve essere inserito in Accumulatore, il dato da scrivere nel registro E.
Output:	Non vengono restituiti parametri interessanti.
Alterazioni:	Nessuna
Nota:	Le interruzioni vengono disabilitate durante l'esecuzione di questa routine, e riabilitate al rientro al programma chiamante.

Nome:	RDPSG
Indirizzo:	&H0096 - 00150
Funzione:	Legge un valore da uno dei registri del PSG.
Input:	Il numero del registro interessato deve essere in Accumulatore.
Output:	Il contenuto del registro viene restituito in Accumulatore.
Alterazioni:	Soltanto l'Accumulatore.
Nota:	Le interruzioni vengono disabilitate ed abilitate all'inizio ed alla fine della routine.

Nome:	STRTMS
Indirizzo:	&H0099 - 00153
Funzione:	Inizia l'esecuzione della musica di sottofondo, eseguita dalla routine di servizio delle interruzioni contemporaneamente all'esecuzione del programma normale, se è stato previsto un buffer con essa.
Input:	Nessuno.
Output:	L'Accumulatore contiene zero solo quando il buffer sonoro è vuoto.
Nota:	La musica di sottofondo inizia realmente ad essere eseguita alla successiva richiesta di interruzione IRQ (generata dallo *hardware* ogni 20 millisecondi).

A cura di Luigi Callegari



Anche queste porte sono identificate da indirizzi, e per non cadere nel tecnico non dico altro.

Negli MSX si utilizzano le porte di I/O per accedere al video (attraverso l'apposito integrato) piuttosto che al Banck Switching ecc.

Queste istruzioni non sono in realtà particolarmente utili al programmatore, perchè come tutti sanno è molto più comodo e più sicuro utilizzare le routines del BIOS che agire direttamente tramite le porte di I/O.

Comunque questo può rendersi talvolta necessario per uno sfruttamento particolarmente efficiente della macchina, legato in special modo alle porte 98H e 99H, utilizzate rispettivamente per la lettura e la scrittura dalla VRAM.

Venendo alla descrizione delle istruzioni in questione, ne esistono due gruppi, il primo per operazioni di I/O di un solo byte e il secondo per blocchi di byte.

Le istruzioni che riguardano la trasmissione di un solo byte si chiamano appunto **IN** e **OUT**, e consentono due formati: il primo è **IN A,(n)** e **OUT(n),A**
dove n indica la porta interessata mentre A contiene il byte da trasmettere (o ricevere), mentre il secondo è:

IN r,(C) e **OUT(C),r**

dove il registro C seleziona la porta mentre r è uno dei registri a otto bit e contiene il dato.

Viste così si potrebbe dedurre che le porte indirizzabili siano solo 256 (e in effetti sugli MSX vengono utilizzate soltanto porte qui comprese): in effetti, però ne possono venire indirizzate 65536 grazie alla capacità dello Z80 di scrivere sugli indirizzi alti il contenuto di:

A per IN A,(n) e OUT(n),A
B per IN r,(C) e OUT(C),n

Per quanto riguarda la trasmissione in I/O di blocchi di dati, il discorso si fa un po' più articolato, analogamente a quanto era successo la scorsa puntata con le operazioni per il trasferimento di blocchi di memoria.

L'indirizzamento avviene per tutte nel seguente modo: il numero della porta nel registro C, l'indirizzo del primo dato in HL e la dimensione del blocco in B (dal che si ricava che un blocco non può superare i 256 byte).

INIR e **OTIR** incrementano HL dopo ogni trasferimento.

INDR e **OTRD** decrementano HL dopo ogni trasferimento.

INI e **OUTI** trasferiscono un solo byte incrementando HL.

IND e **OUTD** trasferiscono un solo byte decrementando HL.

Sottolineo come le ultime due istruzioni trasferiscano un solo byte, ma richiedano di essere trattate insieme alle operazioni per i blocchi perchè hanno lo stesso identico funzionamento ed indirizzamento.

Gli Interrupt

Veniamo ora alla trattazione delle ultime istruzioni riguardanti la CPU, quelle cioè relative all'uso delle interruzioni.

Premetto però che, come ripetuto più volte su queste pagine, è meglio sfruttare gli Hook in RAM che complicarsi la vita con le interruzioni, e perciò qui tratteremo i primi mentre vedremo solo rapidamente le seconde, rimandando al futuro un loro eventuale studio approfondito.

Queste istruzioni sono **IM 0**, **IM 1** e **IM 2** per selezionare le diverse opzioni di interruzioni mascherabili, dalle quali si ritorna al programma principale tramite RETI, che funziona in modo analogo a RET. A voi basti sapere che il modo normalmente selezionato dagli MSX è il modo 1, e che non esistono, come al contrario può avvenire su altri computer con lo Z80, le interruzioni non mascherabili con la relativa RETN.

E' invece molto importante ricordare l'uso di due istruzioni strettamente legate agli interrupt:
EI (Enable Interrupt, abilita interruzioni),
DI (Disable Interrupt, disabilita interruzioni).

Alcune routine del BIOS disabilitano (DI) le interruzioni; è compito nostro ripristinarle (EI) almeno prima di ritornare al Basic.

Ciò che invece risulta grandemente utile alla programmazione è, come abbiamo già detto, il meccanismo degli Hook, che non sono altro che particolari locazioni di memoria logicamente riunite in gruppi di cinque Byte alle quali il microprocessore ogni tanto è costretto a saltare tramite una CALL.

Normalmente però questo salto non provoca alcun risultato, perchè le locazioni contengono dei codici C9H, cioè delle RET, e sono fatte apposta per consentire ai poveri smanettoni di intercettare il lavoro del BIOS, e dato che di 'aggancì ne esistono da FD9AH a FFCAH, le routine che possono essere intercettate sono praticamente tutte le principali.

Cosa c'è in programma...

Per chiarire il modo di utilizzare questa tecnica, ho preparato anche questa volta un programmino abbastanza inutile ma molto didattico, che sfrutta l'aggancio più utile, che viene chiamato ogni cinquantesimo di secondo dalla routine in 38H, che è quella che si occupa di aggiornare il video, il buffer tastiera, il valore di TIME...

E proprio di TIME si tratta, dato che la routine serve semplicemente a produrre un BEEP ogni dieci secondi, mentre nel frattempo è possibile eseguire un altro programma BASIC, o anche in L/M, purchè non utilizzi la stessa zona di RAM della routine o lo stesso aggancio.

Il funzionamento della routine è in realtà semplicissimo: ogni cinquantesimo di secondo controlla se il valore di TIME è pari a 01F4H, cioè cinquecento cinquantesimi, e in caso affermativo emette il famigerato BEEP, azzera TIME e ricomincia daccapo: più facile di così...

Comunque avrete capito che questa routine può diventare il cuore di un programma per fornirci di un orologio simile a quello che hanno a disposizione sistemi operativi più complessi, e non una di quelle routine incredibilmente inutili che per fornirci l' ora, magari con un bel quadrante a lancette, chiedono per sè tutta l'attenzione della CPU.

Oltre al listato Assembler trovate anche il caricatore in Basic.

Digitate semplicemente il programma e mandatelo in esecuzione.

Concludendo...

Con questa puntata abbiamo terminato tutte quante le istruzioni della CPU, ma il corso di Assembler non finisce affatto qui, per l'ottima ragione che le istruzioni per il microprocessore sono inutili se non si sanno gestire tutte le periferiche: a che serve un linguaggio con il quale non si possono gestire il video nè il suono, la tastiera, i joystick, il registratore per salvare i nostri dati, la stampante....??
Comunque vi anticipo che è mia intenzione sorvolare sugli aspetti dell'hardware più risaputi, per attenersi a ciò che concerne le utilizzazioni pratiche, a cominciare dal Generatore di Suoni Programmabile, per gli amici PSG, che è già stato oggetto di una serie di articoli di Lorenzo Carrara, ma che noi esaminaremo fondamentalmente sotto l'aspetto del software.

(List a pag. XXXIII)

ZIG ZAG

di Ercole Murano

Dagli schermi di Canale 5 allo schermo del Vostro MSX

Il programma Zig Zag è la versione computerizzata del "Gioco degli Omini" messo in onda dal famoso Network nella simpatica trasmissione serale "Zig Zag".

Il gioco consisteva in questo: ai concorrenti veniva mostrata per un certo tempo una piramide formata da tanti omini di tre colori differenti; blu, rossi e bianchi.

I giocatori, in questo lasso di tempo, dovevano memorizzare la sequenza dei colori partendo dalla base della piramide.

A questo punto la piramide veniva abbattuta e, a turno, i concorrenti dovevano recitare l'esatta sequenza di colori sino a quando non sbagliavano o raggiungevano la cima della piramide; in questo caso avevano vinto il gioco.

Se invece un colore della sequenza era sbagliato la piramide veniva abbattuta e il giocatore ricominciava da capo.

Le regole per la versione MSX sono identiche ad eccezione del numero di omini costituenti la piramide che possono essere, a scelta, 21, 28 o 36.

Dato il RUN al programma incontrerete una prima schermata di presentazione, una concisa spiegazione del gioco e quindi il menu per la scelta del numero di omini che formeranno la piramide.

Viene quindi mostrata la piramide. Non appena vedrete sparire gli omini potrete iniziare ad inserire la sequenza digitando:

"A" per gli omini blu ("A" sta per Azzurro),
"B" per gli omini Bianchi,
"R" per gli omini Rossi.

Nel caso, assai probabile, che si sbagli la sequenza il computer lo segnalerà acusticamente e mostrerà sulla destra dello schermo l'esatto colore dell'omino.

Nel programma si fa uso di chiamate al BIOS che disabilitano la visualizzazione dello schermo durante la tracciatura delle schermate, quindi non spaventatevi se in alcune fasi non avrete per alcuni secondi nessun riscontro visivo di quanto sta succedendo.

Il Listato

La sequenza di esecuzione del programma passa prima per la subroutine 1210-1320 dove viene disegnata la schermata di presentazione.

La descrizione del gioco avviene fra le linee 120-170 e la scelta del numero di omini che costituiranno la piramide alle linee 210-320.

Nelle linee 360-710 viene disegnato il campo di gioco e mostrata per alcuni attimi la piramide.

Il gioco vero e proprio gira fra le linee 750-980 dove vengono controllati gli input del giocatore, effettuati i controlli e mostrati, a seconda dei casi, o l'omino nella piramide (se il colore è stato indovinato), oppure il colore con cui si sarebbe dovuto rispondere.

La nostra pagella

Il gioco è simpatico e il programma non molto lungo (4700 byte in memoria).

Abbiamo rilevato solo una "piccola" dimenticanza che non abbiamo volutamente corretto:
quando appare per la prima volta la piramide, il giocatore dovrebbe cercare di imparare a memoria la sequenza di colori: ma se chi gioca non è proprio rispettoso delle regole può già da quel momento iniziare a digitare la sequenza, copiandola da quello che vede sullo schermo. che verrà ricordata dal *buffer di tastiera*.

Il gioco finirà in pochi secondi!

(List a pag. XXXIV)

Nella foto i protagonisti del gioco televisivo Zig-Zag, Sandra Mondaini e Raimondo Vianello.

64 COLONNE IN SCREEN 2

di Marco Silvestri

Come aumentare il numero di caratteri per riga.

Uno dei limiti più pesanti all'utilizzo sugli MSX 1 di programmi "seri" quali Word Processor, fogli di calcolo ecc., è certamente quello di avere a disposizione solo 40 colonne di testo.

Un po' poche per un uso efficiente di software che richieda molto spazio sul video; senza contare che qualora sia necessario affiancare ai testi della grafica, si deve ricorrere alla visualizzazione su Screen grafico (256 * 192) e le colonne diventano soltanto 32 e per di più difficilmente gestibili.

Per risolvere completamente questi problemi, sarebbe necessario acquistare un'altra scheda video che consenta le "classiche" 80 colonne, ma si può tentare di ovviarvi parzialmente tramite l'utilizzo di un programma come questo, che sfrutti all'osso i 256 pixel disponibili nel modo grafico.

In che modo? Semplicemente (anche se a scapito della leggibilità), riducendo le dimensioni di ciascun carattere, fino ad arrivare a disporre di 64 caratteri per riga, ciascuno con una matrice di 4 x 8 pixels.

L'aspetto più interessante del programma è però quello di consentirne l'uso da Basic in modo semplice, non richiedendo POKE nè per comunicare la stringa da stampare, nè per indicare la posizione dello schermo desiderata: infatti la stringa viene passata come parametro della USR che richiama la routine, mentre le cordinate X e Y vengono trasmesse tramite una PRESET nel punto desiderato, oppure, per chi non volesse rischiare di alterare con questa istruzione il colore di pixel adiacenti, pokando direttamente in FCB7h per la X e in FCB9h per la Y.

L'altro aspetto interessante è costituito dal fatto che il programma occupa poco meno di 500 bytes di memoria compresi i patterns dei nuovi caratteri (da 54500 a 54988), senza infastidire in alcun modo il buffer del drive; questo risultato è stato ottenuto causando però una certa macchinosità nel programma e un aumento nella velocità d'esecuzione, che risulta comunque irrilevante agli effetti pratici.

Inoltre è possibile utilizzare senza alcun problema i nuovi caratteri insieme a tutti quelli normali e ai grafici in alta risoluzione.

I limiti d'utilizzo sono invece costituiti dal fatto che funziona solo in SCREEN 2 (ma non è possibile fare diversamente) e che nonostante l'uso della PRESET, i caratteri non possono essere stampati a partire da qualunque punto dello schermo ma solo rispettando una griglia di 32 x 24 posizioni come quella dello Screen 1.

Un'ultima limitazione, peraltro facilmente superabile, è che la routine riconosce solo i caratteri con codice ASCII da 32 a 96, e quindi solo le maiuscole.

Per chi lo desidera è comunque possibile operare questa modifica, ricordandosi di ritoccare anche il controllo che impedisce il passaggio al programma di caratteri non previsti.

Per i futuri utilizzatori rimane solo da dire che la routine può essere usata senza timore, perchè anche nel caso di un utilizzo scorretto si limita a provocare una "Illegal function call", senza altri danni per i vostri listati.

Il Listato Assembler

La routine vera e propria occupa l'area di memoria fra 54800 e 54988, mentre i 300 bytes precedenti sono occupati dalle definizioni dei nuovi patterns dei caratteri.

Il suo funzionamento è il seguente: per prima cosa (linee 25-41) calcola la locazione di memoria che corrisponde all'indirizzo al quale iniziare la scritta moltiplicando e sommando opportunamente i valori delle coordinate comunicate tramite PRESET e letti nelle variabili di sistema dove vengono depositati; questa fase viene ripetuta una sola volta per ogni stringa stampata, infatti a partire dall'indirizzo calcolato ora si possono pokare uno dopo l'altro tutti i bytes che costituiscono le parole di una stringa, anche se si prolunga su righe diverse.

Dopo questo si passa a cercare dove la stringa, parametro della USR, si trova in memoria (47-58); questa ricerca avviene secondo le modalità che abbiamo spiegato in una lezione del corso di Assembler (Personal Computer N. 8/86), alla quale rimandiamo chi desideri saperne di più.

La fase successiva (62-67) è la verifica del termine della stringa, che funziona anch'essa tramite le variabili di sistema gestite dall USR.

Si passa quindi a calcolare dove, nella tabella dei patterns, si trovi il primo carattere, si verifica di nuovo il termine della stringa e si cerca quindi il secondo dei caratteri che occupano il byte da stampare (linee 72-107).

Fin qui non c'è niente di difficile, e le difficoltà cominciano solo per la strana maniera nella quale ho memorizzato le sagome dei caratteri: infatti normalmente vengono usate 8 locazioni di memoria per carattere, cioè una per ogni linea, e così in questo caso si avrebbe avuto un ingombro di 64 x 8 = 512 bytes.

Ma siccome in realtà i caratteri sono larghi quattro pixel, è possibile far stare in ogni byte due linee di pattern e occupare in tutto la misera quantità di 64 x 4 = 256 bytes.

Però, mentre è facilissimo pokare le definizioni in questo modo, diventa assai scomodo utilizzarle, perchè si verificano due problemi:

1 - bisogna leggere solo la parte del carattere che interessa, eliminando l'altra;
2 - in questo modo la definizione dei caratteri se ne sta per metà nel nibble alto e per metà nel nibble basso, mentre deve stare tutta nel nibbles alti se il carattere è nella parte sinistra dei bytes interessati, o viceversa nei nibbles bassi.

Considerando che, per esempio, per una stringa di due righe di schermo bisogna elaborare in questo modo 512 bytes, è evidente perchè il programma richiedeva l'uso del linguaggio macchina!

La routine provvede infatti a risolvere i problemi che abbiamo elencato leggendo i bytes, cancellando i nibbles inutili con l' operazione AND e facendo shiftare le sagome nella posizione voluta tramite le operazioni SRL e SLA (112-142).

Al termine vengono incrementati tutti i puntatori, e poi il programma riprende dal primo controllo del termine della stringa (143-152).

CALL 004DH è la chiamata alla routine del BIOS che si occupa di scrivere il valore contenuto in A all'indirizzo specificato da HL (relativo chiaramente alla VRAM).

Il Listato Basic

Si tratta di una routine pensata a scopo dimostrativo, per mostrare come, nonostante i limiti di cui si è detto riguardo al posizionamento delle scritte sul video, sia possibile creare anche qualcosa di abbastanza flessibile come è appunto una routine di input.

Il solo aspetto rilevante è comunque rappresentato dal fatto che le linee 1000-1060 possono essere facilmente copiate per una soubroutine richiamabile da vari punti del programma, ovviando in tal modo ad una delle poche lacune del basic MSX, che è proprio quella dell'input in SCREEN 2.

Per utilizzare invece solamente la possibilità di stampa su 64 colonne sarà sufficiente estrapolare dal listato i seguenti gruppi di linee:
20-170 per la preparazione del linguaggio macchina e dei pattern,
10000-20280 che contengono il linguaggio macchina e la forma dei pattern.

La chiamata alla routine avviene come esemplificato dalla linea 510 (chiaramente dopo aver attivato lo Screen 2).

(List a pag. XXXVI)

AUTOLOADER

di Gianni Bellomusto

Caricare ed eseguire automaticamente qualsiasi programma.

Più di una volta mi sono sentito dire che l'MSX, pur disponendo di un Basic molto versatile e di facile apprendimento per i neofiti della programmazione, diventa un sistema di difficile manipolazione per chi intende usarlo esclusivamente come tastiera da videogioco a causa della multitudine di comandi adibiti al caricamento e all'esecuzione dei diversi file.

Ad esempio i programmi BASIC si possono caricare con *RUN "nome"*
piuttosto che *LOAD "nome",R* , i programmi in BINARIO con *BLOAD"nome",R* , gli schermi con *BLOAD"nome",S*.

E poi ci sono quei file con estensione .COM e .BAT eseguibili solamente in ambiente MSX-DOS.

Non tutti digeriscono questi, sia pur pochi, comandi e comunque con questo programma di Autoload possiamo dare un piglio più "professionale" al sistema. Ovvero, perchè non automatizzare il tutto?

Come usare l'autoloader

Come già avrete capito questo programma permette di mandare automaticamente in esecuzione tutto ciò che è eseguibile cioè programmi BASIC, Linguaggio Macchina e programmi sotto MSX-DOS, nonchè mostrare eventuali disegni grafici. Solo i file di dati (e quindi anche programmi Basic salvati con l'opzione **,A**, ndr) non saranno accettati o intesi come file non eseguibili.

Va subito detto che Autoloader deve essere salvato come **AUTOEXEC.BAS**: così facendo, il programma verrà automaticamente eseguito all'accensione del sistema (chiaramente con il disco già inserito nel drive). Più avanti nell'articolo verrà comunque spiegato il set up del dischetto.

Appena in esecuzione il programma legge e memorizza in un vettore la directory del disco.

Mostrerà quindi su due colonne i nomi dei primi 20 file: con l'aiuto dei tasti cursore si sceglie il programma da caricare e si conferma la sua esecuzione con la pressione del tasto RETURN.

Con F1 e F2 è possibile, quando vi siano più di 20 file, spostarsi da una pagina all'altra della directory.

Ma come vengono riconosciuti i file programma dai file dati?
Quando si richiede l'esecuzione di un file, questo viene aperto come file RANDOM e viene letto il primo byte: a seconda del valore di questo byte, è possibile risalire al tipo di file, Basic oppure Binario.

Nel caso del file Binario occorre leggere anche il secondo ed il terzo byte che combinati insieme danno l'indirizzo di inizio del programma; se questo indirizzo è zero si tratta allora di un file picture, da allocare nella Video Ram.

Purtroppo a questo punto mi sono accorto che non è possibile risalire allo schermo grafico utilizzato dalla figura. Ho comunque optato per lo SCREEN 2: dunque siete avvisati, se caricando un file picture otterrete delle schermate strane ciò vuol probabilmente dire che il file andrebbe in un modo screen diverso dal 2.

Se invece l'indirizzo di start è differente da zero allora si tratta di un file BINARIO caricabile con BLOAD.

E' chiaro che se il programma è composto da:
PIPPO.BAS, PIPPO1.BIN e PIPPO2.BIN
dovrete caricare il... caricatore Basic "PIPPO.BAS" e non uno degli altri due file, pena il concludere un bel niente.

Per i file sotto MSX-DOS il discorso è un po' più complicato.

Come forse sapete se è presente sul disco il file MSXDOS.SYS, un eventuale AUTOEXEC.BAS (in effetti il nostro Autoloader) non viene caricato ne tantomeno eseguito.

Si deve quindi ricorrere ad un trucco: il file MSXDOS.SYS deve essere si presente su disco ma con il nome MSXDOS, senza estensione.

Il sistema operativo, non trovando il file di nome MSXDOS.SYS, manda in esecuzione AUTOEXEC.BAS.

Quando dal nostro programma richiediamo l'esecuzione di un .COM o .BAT, l'Autoloader rinomina MSXDOS in MSXDOS.SYS, scrive un file .BAT contenente il nome del programma da eseguire e resetta il sistema.

Per chiarezza facciamo un esempio passo-passo: supponiamo di voler caricare il programma MED.COM.

Fase 1:
la situazione del disco, relativamente ai soli file che ci interessano, è la seguente:

MSXDOS
AUTOEXEC.BAS
MED.COM

viene caricato AUTOEXEC.BAS, da cui scegliamo di caricare MED.COM.

Autoloader modifica così il disco:
MSXDOS.SYS
AUTOEXEC.BAS
MED.COM
AUTOEXEC.BAT

Il file AUTOEXEC.BAT viene scritto da Autoloader e conterrà le seguenti linee:
rem msxdos.sys msxdos
MED

La prima linea ripristina in effetti la situazione iniziale così che il prossimo autostart ricaricherà ancora l'Autoloader.

Ora viene eseguito il Reset del sistema.

Fase 2:

Il tutto riparte ma questa volta viene incontrato MSXDOS.SYS: si entra nel sistema operativo che esegue quando specificato nell'AUTOEXEC.BAT, e il gioco è fatto!

Il giro è un po' tortuoso, se volete, ma efficace.

Il Listato:

110-150 = preparazione sistema e riconoscimento se MSX1 oppure MSX2.
170-370 = inserimento nomi file in vettori.
510-650 = stampa della videata principale.
690-790 = scelta del programma tramite i cursori e RETURN.
791-795 = se file DOS creazione file AUTOEXEC.BAT.
810-870 = carica file BINARI e BASIC.
890-930 = gestione delle pagine.
970-1010 = gestione degli errori.
1050-1110 = apertura in modo random del file da caricare e lettura dei primi tre byte.

Preparazione del dischetto

Se sul vostro disco troveranno posto solo programmi Basic e Binari (non programmi MSX-DOS) nel disco stesso deve essere presente l'Autoloader con il nome:
AUTOEXEC.BAS

Se sul disco saranno presenti anche programmi che abbisognino dell'MSX-DOS dovrà esserci anche MSXDOS.SYS: copiatelo così:

COPY "A:MSXDOS.SYS" TO "B:"

NAME "MSXDOS.SYS" AS "MSXDOS"

Può sembrare complicato, ma seguendo con attenzione le istruzioni non dovreste incontrare problemi.

(List a pag. XLI)

MUTUI

di Francesco Scarfato

Il programma permette di calcolare il piano di ammortamento di un determinato mutuo secondo un piano mensile, bimestrale, quadrimestrale, semestrale o annuale.

Appena lanciato il programma, sullo schermo appare la schermata di presentazione, rilasciabile premendo SPACE.

Appare il menu principale da cui scegliere il tipo di rateo (annuale, semestrale ecc.); la scelta avviene utilizzando i tasti cursore su e giù per posizionare l'indicatore sul rateo voluto. Posizionandosi su FINE il programma ha termine.

Dopo la scelta del rateo occorre comunicare al programma il numero delle rate, il valore del capitale e l'interesse da applicare.

Nelle schermate successive appare tutto il piano di ammortamento del mutuo.

Otteniamo dati quali l'importo di ogni rata, il valore del capitale, gli interessi e il totale da rimborsare (capitale + interessi).

Nelle righe sottostanti viene mostrata la situazione debitoria (ovvero quanto ancora rimane da versare) dopo il pagamento della n-esima rata.

Qualora il numero di rate sia superiore a dieci, premendo SPACE si può passare alla visione delle rimanenti rate.

Il programma non permette l'uscita su stampante in quanto il sottoscritto non possiede ancora tale periferica. Non è comunque difficile implementare l'uscita su carta dei dati.

Le variabili utilizzate

Fra le variabili utilizzate nel programma citiamo le più significative:
I contiene l'interesse,
K l'importo della rata,
N il numero delle rate,
C il capitale,
H gli interessi maturati,
Z il totale da rimborsare,
A il saldo dopo la n-esima rata,
M il numero dei mesi necessari per il calcolo degli interessi.

La nostra pagella

Il programma appare veramente ben fatto: unico neo non tanto la mancanza della possibilità di stampa su carta dei risultati (la velocità del programma non la rende strettamente necessaria) quanto l'impossibilità di rivedere la situazione debitoria dello schermo precedente.

(List a pag. XLIII)

QUIZ

di Francesco Martinelli

Un semplice, simpatico gioco.

Questo programma, ispirato ad un famosissimo gioco, offre la possibilità di verificare le proprie prestazioni visive e di attenzione.

Si tratta di indovinare una "parola misteriosa" pensata dal computer (o da un giocatore, a scelta).

Un tabellone riporta tutte e 26 le lettere dell'alfabeto; ogni lettera della "parola misteriosa" viene in successione "accesa" sul tabellone.

Il giocatore deve tentare di ricostruire la parola.

Il programma inizia con il chiedere il livello di gioco (da 1 a 10): quanto più è alto il livello scelto, tanto è maggiore il periodo di tempo necessario alla visualizzazione delle lettere della parola, facilitando così il giocatore.

Viene quindi richiesto se la parola vada inserita da un giocatore (nel caso si giochi in due o più): rispondendo "S" verrà richiesta la parola, rispondendo "N" la parola viene "pensata" dal programma.

Si preme quindi un tasto ed appare il tabellone. Ogni lettera viene evidenziata e al termine viene richiesta la soluzione.

Se sbagliamo il programma proporrà la risposta esatta.

Il programma occupa circa 2500 byte in memoria.

Il Listato

100-120 preparazione sistema,
160-190 richiede il livello di gioco,
230-240 chiede l'inserimento della parola oppure,
280-320 la parola viene scelta dal programma,
410-470 viene visualizzato il tabellone (Screen 3),
510-570 viene ricercata la coordinata della lettera da evidenziare,
650-700 viene evidenziata la lettera,
720-810 viene richiesta la parola e mostrato l'esito del gioco,
860-861 DATA contenenti le lettere del tabellone e le relative coordinate,
1080-1090 le parole che possono essere scelte dal programma.

La nostra pagella

Il programma è stato scritto da un giovane di 12 anni, a cui va il nostro plauso e l'incoraggiamente a perseverare nell'uso intelligente che fa del computer, così come ci ha dimostrato.

Al simpatico Francesco vogliamo solo offrire qualche spunto e precisazione senza voler, per carità, stravolgere l'algoritmo del suo Programma.

Innanzitutto avrà sicuramente notato che abbiamo tolto alcune istruzioni ridondanti (quali la più volta ripetuta apertura del canale verso lo Screen 3).

Volevamo poi ricordare che nel Basic MSX soltanto le prime due lettere del nome di una variabile sono importanti: così nelle linee 160-190 quando fai riferimento alle variabili LIV e LIVE in effetti ti riferisci sempre e solo a LI: nello specifico caso non ci sono controindicazioni, ma tieni questo fatto sempre in massima considerazione.

L'algoritmo delle linee 280 e 290 avrebbe potuto essere scritto in un unica istruzione:

```
RN=INT(RND(-TIME)*35)+1
```

e visto che in effetti RN ti serve una sola volta, mettere questo calcolo dopo il TO del FOR:

```
FOR N=1 TO INT(RND(-TIME)*35)+1
```

Sempre per ottimizzare e velocizzare il programma, sarebbe stato bene inserire le coordinate in un vettore bidimensionale (ad esempio CO(0,0)=32:CO(0,1)=40:CO(1,0)=64 etc.) caricato una volta per tutte con un ciclo FOR READ NEXT e calcolare l'indice dell'array direttamente dal codice ASCII della lettera da evidenziare nel tabellone: ad esempio sottraendo 65 dal codice ASCII della "A" 65 ottieni 0, sottraendolo dalla "B" ottieni 1 e via di seguito;

```
AS=ASC(MID$(PA$,N,1)-65)

YA=CO(AS,0):YB=CO(AS,1)
```

Facci sapere se sei d'accordo con questi suggerimenti...e continua così che vai forte!

(List a pag. XLV)

USA & ABUSA

a cura di Davide Serioli

Semplici routine e piccoli trucchi per tutte le tasche.

USA & ABUSA tratta di ideuzze per programmare meglio, per sfruttare la macchina più a fondo, subroutine multiuso da inserire nei vostri listati. Naturalmente con il contributo di voi lettori.

Alcune note per l'uso: il numero di linea 0 (zero) indica che la linea deve essere una delle prime nel vostro programma: gli altri numeri di linea (da 1 in su) hanno il solo scopo di evidenziare salti e di raggruppare la subroutine. Dove non vedete numeri di linea viene sottointeso che queste linee possono essere poste dove vi servono o prima di chiamare la subroutine. Qualora si parlasse di routine che girano solo su MSX 1 o MSX 2 verrà chiaramente annotato.

Se avete problemi, domande, migliorie su una routine specificate il numero e il nome della routine. Grazie, ciao, buon lavoro!

P.S.: gli Usa & Abusa dal 25 al 28 sono del Sig. Francesco Scarfato di Gragnano (NA).

MailBox

Dal tenore delle lettere che riceviamo, sembra che la rubrica Usa & Abusa abbia suscitato un certo interesse fra i lettori.

Ricordiamo che la rubrica si basa soprattutto sul vostro contributo e quindi non esitate a mandarci le piccole scoperte che avete fatto utilizzando gli MSX.

Fra le altre è giunta in redazione anche la lettera di un attento lettore, il Sig. Giancarlo Piacenza di Verzuolo (CN), che segnala migliorie e considerazioni su alcuni Usa & Abusa apparsi sul numero 13 di Personal Computer.

Ecco quanto ci segnala il Sig. Giancarlo:

17) VDP ON/OFF

La linea 0 contiene un refuso tipografico e va riscritta nel seguente modo:

```
0 DEFUSR=&H44:DEFUSR1=&H44
```

Il VDP ON/OFF si può ottenere, e più facilmente, con le istruzioni:

```
VDP(1)=VDP(1) AND 191
```

per disattivare il video,

```
VDP(10=VDP(1) OR 64
```

per riattivarlo.

Il bit numero 6 del registro 1 del VDP serve appunto a predisporre l'area di schermo attiva: la prima istruzione ponendo a zero tale bit fa assumere a tutto lo schermo il colore del bordo, mentre la seconda, forzando il bit a uno, attiva lo schermo.

23) NO LIST

Non è necessario dare la POKE &HFF07,201 in quanto la locazione appartiene alla zona degli Hook e quindi contiene già tale valore in permanenza.

(Vorremmo far notare che gli Hook non contengono sempre e permanentemente un 201: la presenza o meno di tale valore dipende dalla configurazione del sistema, Ndr)

25) Nuovi caratteri 2

Con queste poche linee è possibile definire un nuovo set di caratteri per gli schermi in modo testo (0 a 40 o 80 colonne, 1 a 32 colonne). Il set che otterremo sarà simile a caratteri in grassetto.

```
1 DEFINT A-Z
2 SC=?
3 FOR I=BASE(SC)+32*8 TO BASE(SC)
+126*8
4 VPOKE I,VPEEK(I) OR VPEEK(I)/2
5 NEXT
```

Per la variabile SC inserite 2 per lo SCREEN 0 oppure 7 per lo SCREEN 1.

26) Nuovi caratteri 3

In questo ennesimo set, ogni carattere viene ridefinito tramite valori contenuti nei DATA che vengono sequenzialmente letti e "Pokati" nella VRAM.

Il programma, essendo relativamente lungo, appare nelle pagine riservate ai listati.

Da far notare che il set, seppur pensato per lo Screen 0, può essere adattato allo Screen 1; basta modificare le seguenti linee:

```
210 SCREEN 1
220 WIDTH 32
270 B2=BASE(7)+31*8
```

Non tutti i caratteri vengono redefiniti, ma soltanto quelli compresi fra i codici ASCII 32 e 122. Per capire a quale carattere facciano riferimento le varie linee DATA basta guardare il numero di linea: esso rappresenta il codice ASCII del carattere.

27) RESTORE KEY

Se avete pasticciato con i tasti funzione e volete ripristinare le stringhe di default potete semplicemente usare:

```
DEFUSR=&H3E:A=USR(0)
```

Sino al momento in cui non eseguite un CLS o un KEYON oppure non premete lo SHIFT sullo schermo appariranno i vecchi assegnamenti anche se i default sono stati già ripristinati.

28) PSET senza PSET (MSX 1)

Se siete in pagina grafica e utilizzate PSET o PRESET per posizionare il cursore, per evitare la comparsa di qualche antiestetico punto non desiderato potete inserire direttamente i valori X e Y nei due byte che controllano la posizione del cursore:

```
POKE &HFCB7,X
POKE &HFCB9,Y
```

29) PSET senza PSET (MSX 2)

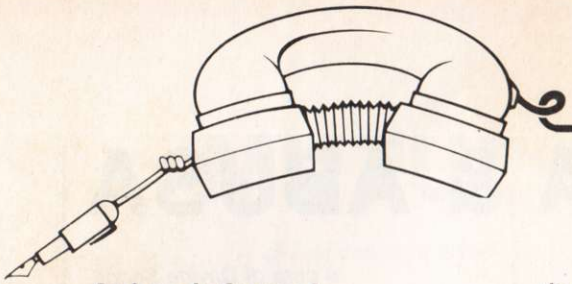
Sempre relativamente al problema di cui alla numero 27, se desiderate utilizzare l'idea anche sugli MSX 2 occorre fare qualche considerazione:

- se vi trovate ad operare negli SCREEN 2, 3, 4, 5 o 8 tutto funziona come già detto;
- se invece siete negli SCREEN 6 o 7 (che permettono una risoluzione orizzontale di 512 pixel) dovrete "Pokare" due byte e precisamente &HFCB7 ed &HFCB8:

```
POKE &HFCB7,X mod 256
POKE &HFCB8,X / 256
```

Queste locazioni fanno chiaramente riferimento solo alla pagina attiva.

(List a pag. XLVI)



mailbox

Scrivere in Screen 4-

Sono un ragazzo di 13 anni possessore di un computer Philips Msx 2 NMS 8280 e vi scrivo per porvi alcune domande che ritengo di interesse generale. Nell'ordine:

1) digitando ERROR 60 sullo schermo appare la scritta "Bad FAT": cosa significa?

2) digitando REMUN sul computer non appare alcun messaggio d'errore eppure consultando il mio manuale di riferimento risulta che tale istruzione non esista. Posso avere chiarimenti?

3) Sull'MSX 2, negli schermi grafici "nuovi", quando scrivo qualche frase dietro ogni lettera appare uno "sfondo" che copre ciò che si trovava sotto, cosa che non avviene negli Screen 2 e 3: come è possibile ovviare a questo inconveniente?
Vi sarei anche grato se voleste pubblicare il mio indirizzo in quanto desidero fondare un Club Msx nella zona di Salerno.
Un caro saluto.

Alfonso Florio
Via Matierno 5
84100 Salerno
tel. 089/271300

1) Su ogni disco viene creata una mappa relativa ai settori occupati dai vari file presenti. Tale mappa viene chiamata FAT (File Allocation Table, Tabella di Allocazione dei File).
Se questa mappa risulta in qualche modo danneggiata (ad esempio a causa di un DSKO$ utilizzato male o altre manovre errate) si incorre nell'errore 60. Un gran brutto errore: difficilmente i file potranno essere recuperati e il dischetto può venire riutilizzato solo dopo la riformattazione.

2) Se non hai sbagliato a scriverci l'istruzione, REMUN non è altro che una REM seguita da UN. O ti riferivi a qualcosa d'altro?

3) Ogni istruzione grafica riferita ai nuovi modi grafici degli MSX 2 ha anche un parametro detto Operazione Logica.
Questo parametro può essere OR, XOR, AND, PSET e PRESET ecc. e a seconda dei casi esegue una particolare operazione logica fra il colore da te scelto e quello presente sullo schemo.
Se il parametro viene omesso, per default viene utilizzata la PSET con il risultato da te ben conosciuto.
Utilizzando OR lo specifico problema da te lamentato viene eliminato. Ad esempio:

```
PRESET(100,100),,OR
PRINT #1,"CIAO"
```

Ho il rilocatore!

Sono un assiduo lettore di MSX Personal Computer a cui rivolgo i miei complimenti: ritengo sia l'unica che si sta interessando seriamente dei computer MSX.
Ho da porvi un paio di domande:

1) è possibile collegare la stampante VW 0030 della Philips ad un computer dotato di interfaccia seriale o parallela?

2) il disk drive dell'MSX può essere collegato ad un computer della serie Atari ST facendo uso di qualche cavo al posto dell'interfaccia usata sui computer MSX? se si, quale?
Ringrazio anticipatamente sicuro di una vostra celere quanto dettagliata risposta.
P.S.: Sono in possesso di un programma rilocatore in L/M e quindi disposto a cederlo ai lettori.

Vincenzo Maccuro
Via C.C. Sforza 14
70043 Monopoli BA

Ringraziamo per i complimenti e subito passiamo a rispondere alle domande formulate:
1) la stampante è di tipo parallelo Centronics e quindi collegabile a qualsiasi computer dotato di tale interfaccia. Va ricordato che il set di caratteri della stampante rimane quello degli MSX.

2) qui è più difficile dare una risposta precisa. Dalla documentazione in nostro possesso ambedue i disk drive hanno linee di collegamento simili. Purtroppo non conosciamo i dati elettrici di tali linee che è il dato più importante per la fattibilità del collegamento. Inoltre il connettore utilizzato sugli Atari ST e molto particolare e probabilmente di difficile reperibilità.
Quindi piuttosto che rischiare di rompere qualcosa e visto il basso costo dei drive Atari, ci sentiamo di consigliarti l'acquisto di un secondo drive per ST: è probabilmente la scelta più economica.
Sappi anche che un disco formattato dagli MSX può essere tranquillamente letto e utilizzato dagli ST. Non è vero il contrario e quindi se hai necessità di scambiare dati fra i due computer formatta il supporto sugli MSX.

Non ho il rilocatore...

Ho sentito la necessità di scriverVi dopo alcune considerazioni maturate leggendo la Mailbox MSX.
Sono in possesso di un Sony Msx 2 HBF700 da quasi un anno. Come potete immaginare da subito fui alle prese con la famosa "incompatibilità", riveltasi poi una sciocchezza grazie ad un amico smanettone.
Ora quello che trovo inconguente da parte degli utenti MSX è che da una parte si dispiacciono che la diffusione dello standard non sia delle dimensioni che meriterebbe e dall'altra insistano su incompatibilità insuperabili se non con rilocatori vari in L/M che però tengono per sè. Tutto questo, mi pare, genera solo confusione.
Con questo invito i lettori utenti di MSX a mettere a disposizione le loro esperienze, rinunciando a pretese speculative: solo così si contribuisce alla diffusione dello standard, viceversa a cosa servirebbero tante esperienze su uno standard desueto.
Benvenuta quindi la rubrica Usa & Abusa con tanti piccoli suggerimenti e consigli: questo è quello che serve, non la confusione.
Infine un incoraggiamento ed un plauso a Davide Serioli per il lavoro che ci mette a disposizione.
Ho ricopiato il programma di Hard copy e non capisco perchè vengano stampati quei caratteri sulla destra della copia. Allego un esempio e il listato. Potete aiutarmi? *Renato Balbo - Santorso, VI*

Siamo d'accordo con te sul fatto che le esperienze, pur piccole che siano, vadano scambiate fra tutti gli utenti. Purtroppo, come per ogni cosa, nella schiera dei veri appassionati fanno capolino anche persone che in verità hanno solo interesse alla lira con tutte le conseguenze che si creano.
Per venire al tuo problema nel listato che ci hai inviato abbiamo riscontrato due errori di copiatura:
il primo alla linea 10050: la variabile è YY% e non Y% come da te digitato;
il secondo, che è poi quello che genera l'errore nella Hardcopy grafica, si trova alla linea 10210: il ciclo deve terminare a BUFFER+255 e non a BUFFER+256.

UN NUOVO MSX COMPUTER CLUB A GENOVA

Abbiamo il piacere di informarVi che a Genova è operante da poco più di due mesi l'**MSX COMPUTER CLUB**.
I soci fondatori del Club sono cinque e gli aderenti già più di un centinaio, sparsi in tutta Italia.
Il nostro Club ha lo scopo di informare gli aderenti circa le novità di Hardware e Software, di scambiare qualunque informazione utile al migliore utilizzo dello standard MSX e da la possibilità di ottenere i programmi desiderati scegliendoli in una vastissima biblioteca-softeca.
L'iscrizione è gratuita. Per ogni informazione scrivere a:

Filippi Dino
Casella postale 3006
Genova P.P.

IMPORTATRICE DISTRIBUTRICE
IN ESCLUSIVA PER L'ITALIA

ETP SRL
ELECTRONIC AND TECHNICAL PRODUCTS

via del macao, 4 - 00185 roma
tel. 4743080 - 4755875

RITIRIAMO IL VECCHIO
CONSEGNAMO IL NUOVO

Una volta... nel branco...

studio A R T 06/490568

Oggi

EXPRESS
PERSONAL COMPUTER

XT
- Totalmente compatibile PC/XT IBM
- microprocessore Intel 8088
- memoria RAM 256 Kb espandibile a 640 Kb sulla piastra base
- memoria ROM 8 Kb (BIOS) espandibile 64 Kb
- scheda madre con 8 slot di espansione
- scheda grafica monocromatica ad alta ris (720x348 punti)
- porta parallela per collegamento stampante
- tastiera italiana ASCII con 84 tasti e funzione o USACII
- uno o due drive slim 5"1/4 da 360 Kb (o disc di diverse capacità)
- alimentatore 135 W - 220 Volt
- dimensioni: 500 x 410 x 142 mm
- peso: 11 Kg
- sistemi operativi supportati: tutti que PC/XT IBM

AT
- Totalmente compatibile PC/AT IBM
- microprocessore Intel 80286: trasferimento dati a 16 bit; indirizzamento a 24 bit
- memoria RAM 512 Kb espandibile a 3 MB
- scheda madre con 8 slot di espansione
- scheda grafica mono...

franchising MERCATINO® sempre vendite usate
tel: 0678147004
22358 / 624083
RM12
RIVISTE PC
27.06.2017 €4,00 2,00

PERSONAL COMPUTER PC 1640

OGNI SOLUZIONE

E' AD ALTA DEFINIZIONE

Il primo Personal Computer professionale che può permettersi contemporaneamente un'altissima risoluzione grafica, sia in b/n che a colori, e un prezzo veramente accessibile.
Risoluzione a colori EGA di 640×350 punti o Hercules 720×350 in b/n. Superveloce (CPU 8086 a 8 Mhz) ma semplice da usare, PC 1640 Amstrad è compatibile MS-DOS, in grado cioé di utilizzare la più ampia libreria di programmi attualmente in commercio.
PC 1640 Amstrad è disponibile nelle versioni:

PC 1640 SD-MD
b/n, Singolo Drive 360K L.999.000+IVA

PC1640 DD-MD
b/n, Doppio Drive 360K L.1.249.000+IVA

PC1640 HD-MD
b/n, Hard Disk 20Mb L.1.999.000+IVA

PC 1640 SD-CD
col., Singolo Drive 360K L.1.349.000+IVA

PC 1640 DD-CD
col., Doppio Drive 360K L.1.599.000+IVA

PC 1640 HD-CD
col., Hard Disk 20Mb L.2.349.000+IVA

PC 1640 SD-ECD
col., Alta Def., Singolo Drive 360K L.1.599.000+IVA

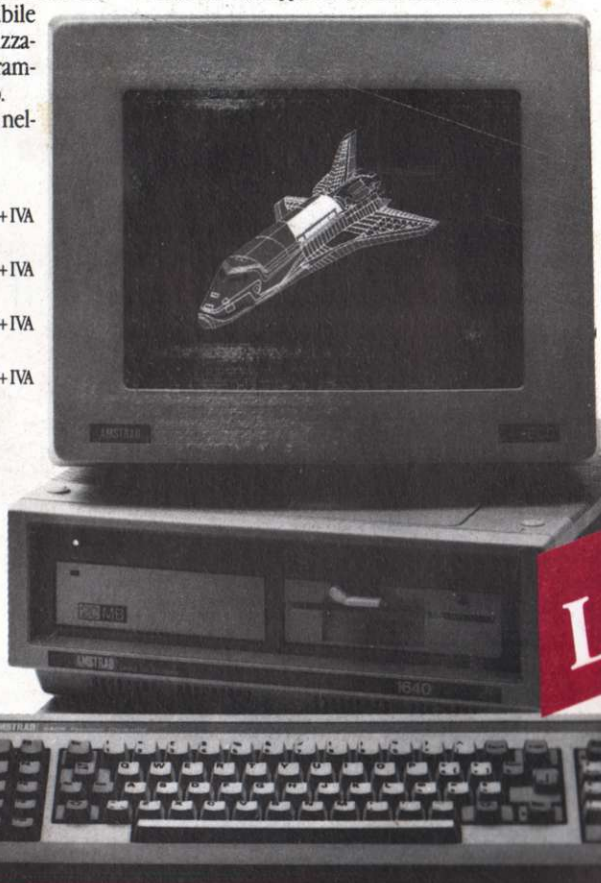
PC 1640 DD-ECD
col., Alta Def., Doppio Drive 360K L.1.849.000+IVA

PC 1640 HD-ECD
col., Alta Def., Hard Disk 20Mb L.2.599.000+IVA

Video grafico, Tastiera, Mouse, RAM 640K, Software MS-DOS e GEM inclusi.
Prezzo e qualità assicurati da oltre 700 punti di vendita e 72 centri di assistenza specializzata. Garanzia 1 anno.

I prodotti Amstrad sono disponibili presso i migliori Computer Shop, le catene Expert (pag. gialle, cat. elettrodomestici), EHP-SINGER (02-646781), per l'industria presso Silverstar (02-4996) e Claitron (02-3010091).

AMSTRAD



A partire da L.999.000*

A — Monitor standard

A — Monitor ad alta definizione.

Fotografia originale

*IVA esclusa

Compatibile EGA / HERCULES / MDA / CGA

1640

Per informazioni inviare a: AMSTRAD S.p.A. BUSINESS DIVISION 20156 MILANO - Via Riccione, 14 - Tel. 02/32.70.741 (ric. aut.)

Nome _____ Cognome _____ Soc. _____

Via _____ Cap. _____ Città _____ Prov. _____ Tel. _____